BALLISTIC RESEARCH LABORATORIES

Report No. 615

A Study of Inverse Interpolation of the Eniac

HASKELL B. CURRY and WILLA A. WYATT

ABERDEEN PROVING GROUND, MARYLAND

RESTRICTED

BALLISTIC RESEARCH LABORATORIES

REPORT NO. 615

A Study of Inverse Interpolation of the Eniac

HASKELL B. CURRY and WILLA A. WYATT

19 August 1946

ABERDEEN PROVING GROUND, MARYLAND

TABLE OF CONTENTS

BALLISTIC RESEARCH LABORATORIES

REPORT NO. 615

Curry/Wyatt/ejs
Aberdeen Proving Ground, Md.
19 August 1946

A STUDY OF INVERSE INTERPOLATION OF THE ENIAC

ABSTRACT

A study is made of methods of using the Eniac to convert a table giving values of two functions **x**(t,∅), for evenly spaced values of t and ∅ into a table giving t for evenly spaced values of **x** and ∅. At the same time there may be additional functions of t and ∅ to be expressed in terms of **x** and ∅. A basic scheme for doing this is set up and programmed in detail. This scheme contains three additional functions y, z, w. The basic scheme is such as to be readily modified; some modifications are studied and programmed in somewhat less detail than the basic scheme.

OUTLINE

LIST OF FIGURES

(With reference to paragraph numbers in text)

INTRODUCTION

1.1 The problem of inverse interpolation may be stated as follows. Suppose we have a table giving values of a function x(t), and possibly some additional functions, for equally spaced values of the argument t. It is required to tabulate t and the additional quantities for equally spaced values of x.

1.2 This problem is important in the calculation of firing tables. Suppose the trajectory calculations have given us the coordinates (x,y) of the projectile as functions of t (time) and φ (angle of departure.) For the tables we want t and φ as functions of x and y; indeed we wish to determine φ so as to hit a target whose position (x,y) is known, and t is needed for the fuze setting or other purposes. This is a problem of inverse interpolation in two variables; it can be solved by two successive inverse interpolations on one variable.

1.3 In this report the problem of inverse interpolation is studied with reference to the programming on the Eniac as a problem in its own right. A basic scheme of programming is set up in detail in such a way that it can readily be modified to suit circumstances. Some modifications are also programmed in somewhat less detail than the basic scheme, and general principles regarding modifications are discussed. However no attempt is made to deal exhaustively with these modifications. The special requirements of firing table calculations, except in so far as they furnish motivation, are not gone into here; a separate report on that subject is planned.

1.4 The process of inverse interpolation is essentially a combination of direct interpolation with solution of an algebraic equation. Thus if we let u be the independent variable, after a suitable change of scale and of origin, and let f(u) be an interpolatory approximation to a given function x(t), the inversion of x(t) is essentially equivalent to solving the equation.

$$f\,(u) = a$$

for a succession of equally spaced values of a. This is done here by an iterative process. The advantages of an iterative process are: 1) that it is eminently suitable for the Eniac, and 2) the process is in principle independent of the choice of the formula for f(u). In order to make a detailed program it is necessary to make a specific choice of f(u); but the use of a different f(u) is a special modification of the sort mentioned in the preceding paragraph.

1.5 In order for the process to have a unique solution we must suppose that

$$\dot{x} = \frac{dx}{dt} \neq 0$$

within the range considered. Here we shall suppose that

$$\frac{dx}{dt} > 0.$$

The case $\frac{dx}{dt} < 0$ can be taken account of by changing the sign of either x or t but not both. For some purposes we shall also suppose that

$$\ddot{x} = \frac{d^2x}{dt^2} < 0.$$

The case that $\ddot{x} > 0$ can be reduced to this by changing the sign of both x and t[1]. Some but not all of the conclusions, however, are valid even if $\ddot{x}$ changes sign. These assumptions of course apply strictly to the x(t) as represented by the interpolation formula f(u). We really suppose the function such that all the f(u) (for the various t_k) are such that the above inequalities apply to the function they represent.

1.6 The above assumptions suffice for the discussion of general principles. For the detailed programming more specific assumptions are necessary, as follows.

1.7 For the present basic scheme the approximation f(u) is formed by ordinary interpolation with third order differences; i.e. f(u) is a cubic polynomial through four consecutive points, two on either side of the point desired. There are three additional functions, called **secondary functions** and designated y(t), z(t), and w(t) respectively, which it is desired to tabulate as functions of a. The same interpolation formula is to be used in calculating these secondary functions by direct interpolation as was used in par. 1.4. The quantities x(t), y(t), z(t), and w(t) are given for evenly spaced values of t on a file of cards with one card for each value of t; the cards are arranged in order of increasing t. Thus it is not necessary to set these functions on the switches. The output is to be a file of cards containing t, y, z, w for each value of a, one card for each value of a. If these quantities are ten-figure numbers the three secondary functions represent the maximum capacity of the machine under the circumstances mentioned. For four successive values of each function are needed; one of these values is on the card in position, and hence is stored in the constant transmitter; the other three must be stored in accumulators. This makes twelve accumulators tied up for storage; since four accumulators must be free in order to operate the multiplier, and four more are needed for quantities to be printed, the Eniac is jammed full. In actual practice we may need less than three secondary functions (thus for the main part of the firing tables we may need only x and y), or we may need more (e.g. for differential effects); in the former case extra accumulators will be available for elaborating certain aspects of the computation as discussed below, or for a higher order of interpolation; in the latter case it will be necessary to have two or more cards for each value of t (see sec. 10).

1.8 In some problems, for instance in the calculation of firing tables, it is desired to perform the inverse interpolation on functions of two variables, the inverse interpolation being with respect to one variable while

[1] These changes in sign of x and t can of course be accomplished in the scheme below by simply changing the sign of Δa and Δt respectively. There is no other change necessary. The different possibilities are as in the following table:

$\dot{x}$	$\ddot{x}$	Δa	Δt
+	-	+	+
-	-	+	-
+	+	-	-
-	+	-	+

If the signs are as in this table, the theory for $\dot{x} \geq o, \ddot{x} < o$ applies. A brief discussion under the assumption $\dot{x} > o, \ddot{x} > o$ is given in pars. 3.23f.

the other is held constant. Here this second variable is designated φ with the understanding that it is not necessarily the angle of departure for a trajectory. Then we have a file of cards and a separate process of interpolation for each value of φ. Let us call such a file of cards with constant φ a **group.** It is then desirable to so program the calculation that the groups can be combined into a single file, and that, on finishing with one group, the machine will automatically make the necessary adjustments and proceed with the next group. This refinement is incorporated in the present basic scheme. It is supposed that the cards in each group satisfy the assumptions in the preceding paragraphs--in particular that they are arranged according to increasing t--; and that the groups are arranged in order of increasing φ .

1.9 Further, more specialized assumptions are as follows. The problem requires several constants, viz.: the intervals Δt, Δa, Δb between successive values of t, a, and b respectively (where a represents x and b represents $\emptyset$ as the arguments of the output); the initial values t_o, a_o, b_o of these same three variables; and a constant λ in the successive approximation scheme. All these constants we suppose to be with only one significant digit, so that in order to multiply by one of these numbers it is sufficient to use a shifter and a repeat switch. (This applies particularly to λ and Δt; cf par. 7.6). Again we suppose there are at least two cards in every group preceding the first value of a; and also two cards following the last value of a. (On what happens if this is not true see par. 11.4 (c)).

2. GENERAL OUTLINE OF THE CALCULATION

2.1 As explained in the introduction we suppose that the input cards give the four quantities, x, y, z, w, as functions of t and φ for equally spaced values of t and φ . The cards are supposedly sorted into groups, in each of which φ is a constant and the cards are arranged in order of increasing t; the groups are then arranged in order of increasing φ .

2.2 It will be convenient to adopt a certain terminology here. The variable φ will be known as the **parameter;** the variable t as the **argument** or, when it is desired to be specific, the **in-argument.** The argument of the output, or **out-argument,** is of course x; but it is convenient to call it a in connection with the output, so that x is reserved for the input function. Likewise it will be convenient to designate the parameter φ , when considered with reference to the output, by a special symbol b; in such a case φ would be called the **in-parameter,** b the **out-parameter.**

2.3 We consider the cards of a single group. Let t_k, x_k, y_k, z_k, w_k be the values appearing on the (k+1)st card of the group (so that t_o, x_o, y_o, z_o, w_o, are the initial values).
Then

$$t_k = t_o + k \Delta t. \tag{2.1}$$

Suppose now that

$$x_k \leq a < x_{k+1}. \tag{2.2}$$

Then we can set

$$u = \frac{t-t_k}{\Delta t} . \tag{2.3}$$

The main problem of inverse interpolation, viz., the determination of t_a so that

$$x\,(t_a) = a,$$

can be regarded as the solution of

$$F(u) = f(u) - a = 0, \tag{2.4}$$

where f(u) is the polynomial approximation to $x(t_k + u\,\Delta t)$.

According to the introduction f(u) is the third degree polynomial such that

$$\begin{aligned} f_{-1} &= f(-1) = x_{k-1}, \\ f_o &= f(o) = x_k, \\ f_1 &= f(1) = x_{k+1}, \\ f_2 &= f(2) = x_{k+2}. \end{aligned} \tag{2.5}$$

This can be represented, in a form convenient for Eniac computation, thus:

$$6f(u) = 6A+(u+1) \left[6B+u \, [3C+(u-1)\,D] \right] , \tag{2.6}$$

where (in the notation of advancing differences)

$$\begin{aligned} A &= f_{-1} \\ B &= \Delta f_{-1} = f_o - f_{-1} \\ C &= \Delta^2 f_{-1} = f_1 - 2f_o + f_{-1} \\ D &= \Delta^3 f_{-1} = f_2 - 3f_1 + 3f_o - f_{-1} \end{aligned}$$

2.4 By our hypotheses the equation F(u) = f(u) -a = 0 will have a unique solution between 0 and 1. We shall call this solution u_a. The corresponding values of t, x, y, z, w, will also be indicated by the appropriate letter with subscript a; thus $x_a = a$ and $t_a = t_k + u_a \Delta t = t_{k+2} + (u_a - 2)\Delta t$. The quantities to be printed are then a, t_a, y_a, z_a, w_a, as well as φ and auxiliary constants (card numbers etc.)

2.5 The calculation relating to a single value of a will be called a **round.** The first job in a round is to read cards until (2.4) is satisfied, with proper safeguards to change the value of a or to start a new group. This will be called the **set-up.** The next main job is to calculate u_a; this will be called the **primary interpolation.** When this is completed it remains to calculate y_a, z_a, and w_a; these interpolations, which are direct, will be called **secondary interpolations**. They are supposed to use the same interpolation formula as the primary interpolation. Finally there is the printing and the clearing and readjustments necessary to conclude the round. These last, which are simple, are conveniently treated as part of the last secondary interpolation. The primary interpolation can be advantageously divided into two parts: the first or **preparatory part** consists of the calculation of the coefficients of f(u) and other items which are the same in all the approximations; the second part, or the **iteration** consists of repetitions of what is essentially the same process to give the successive approximations themselves.

2.6 The entire computation procedure can thus be divided into certain major parts which are repeated over and over according to the programming. These major parts will be called **processes**, and will be designated by Roman numerals as follows:

I. The set up.
II. Preparatory part of the primary interpolation.
III. Iteration.
IV. Secondary interpolation for y.
V. Secondary interpolation for z.
VI. Secondary interpolation for w and closure of the round.

2.7 Before proceeding to details it is necessary to discuss certain matters theoretically. The theoretical discussion of Process III, which forms the heart of the computation, will occupy section 3. The discriminations necessary in Processes I and III involve some matters of principle which will be discussed in section 4. Finally the formation of the coefficients in II, IV, V, and VI proceeds by a special case of some general theorems which may be of interest on their own account; these are the subject matter of section 5.

2.8 The detailed program will concern us in sections 6-7. However since it will be expedient to refer to some of the programs worked out as illustrative examples in the theoretical discussions of sections 4 and 5, it will be necessary to make a general explanation of the technique at this point. Each process is broken into pieces, called **stages**, which are units in the following sense. Each stage is a program sequence with an input and one or more outputs. The input of each stage comes from the output of one or more other stages of the same or different processes; the outputs all go to the input of some other stage or are blank. Otherwise there is no program interconnection between the stages, although stages may run simultaneously and the transmission and reception of the same quantity may be programmed in separate stages. It follows that if one knows what actions the various stages order and how they are connected to one another, he can determine the course of the whole computation without regard to the programming within the stages. The stages can be programmed as independent units, with a uniform notation as to program lines, and then put together; and since each stage uses only a relatively small amount of the equipment, the programming can be done on sheets of paper of ordinary size.

2.9 The interconnection of the various stages will concern us in section 6; here we are interested only in the programming technique for the details. Instead of seven-column computing paper used in a previous report, ordinary 20 x 20 coordinate paper is used with a rectangle 1/2" x 1" for each item. The location of the various indications in the programming is shown in the Code for Programming Conventions in Fig. 35.

The following special conventions will also be used. A line drawn horizontally across the column for an accumulator indicates clearing. A horizontal line with gaps across the column for a stepper is to indicate the positions of the stepper; the gaps indicate the positions which are possible at the step in question. A double horizontal line across a column indicates the column is to be used from that point on for a different piece of equipment; the nature of the new equipment shown by a new heading written immediately under the double line. On some schedules portions of the program which repeat as a unit are enclosed in boxes formed by dashed lines.

2.10 The programs for the separate stages in the basic scheme are given in Figs. 7-26; the schedules for fitting these together in Figs. 28-34. A flow-chart showing the interconnection of the various stages is given in Fig. 44.

3. THEORY OF THE ITERATION

3.1 We are concerned with the solution of the equation (2.4), viz.

$$F(u) = f(u) - a = 0$$

under the assumption that $F(0) \leq 0$, $F(1) > 0$, $F'(u) > 0$ throughout the interval from 0 to 1. The special assumptions of section 1, in particular the fact that f(u) is a cubic, will not be used in this section.

3.2 The equation (2.4) can be put into the form

$$u = G(u),$$

which is suitable for an iterative process, if we set

$$G(u) = u - \lambda F(u),$$

where λ may be a function of u. The corresponding iterative process is

$$u_{n+1} = G(u_n) = u_n - \lambda_n F(u_n). \qquad (3.1)$$

It is clear that if this process converges, say

$$\lim_{n\to\infty} u_n = u_a,$$

and if λ_n converges to a limit λ_a not zero,
then

$$u_a = u_a - \lambda_a F(u_a),$$

and hence

$$F(u_a) = 0. \qquad (3.2)$$

Thus the convergence, if any, is to a root of the equation (2.4)

3.3 Many of the well known processes are special cases of the iterative scheme (3.1). Thus if

$$\lambda_n = \frac{1}{F'(u_n)}$$

the process is known as Newton's method. On the other hand if

$$\lambda_n = \frac{u_b - u_n}{f(u_b) - f(u_n)}$$

where u_b is a point such that $f(u_b)$ and $f(u_n)$ have opposite signs, the iteration is essentially the "Rule of False Position" (Whittaker, E. T. and Robinson, G., **The Calculus of Observations,** Blackie and Son, 3rd edition, 1940, sec. 49).

3.4 The process has also a simple geometrical interpretation. In the (u,x) plane--with u as abscissa, x as ordinate--the equation of a line of slope μ through the point $(u_n, F(u_n))$ is

$$x - F(u_n) = \mu(u - u_n),$$

i.e.

$$u = u_n + \frac{1}{\mu}(x - F(u_n)). \tag{3.3}$$

The intersection of this line with the axis of abscissas is u_{n+1} if $\mu = 1/\lambda_n$. Thus to locate u_{n+1} we draw a line with slope $1/\lambda_n$ through the point on the curve $x = F(u)$ for which $u = u_n$; the intersection of this line with the axis of abscissas is then u_{n+1}. In the following it will be convenient to call the line (3.3) the **approximating line** and to understand that the terms **curve** and **tangent** refer to the curve $x = F(u)$.

3.5 We shall now investigate analytically the general conditions on λ_n in order that the process (3.1) may converge. In this we make use of the fact that there is a unique u_a in the interval $0 \leq u < 1$ for which

$$F(u_a) = 0.$$

Hence we have

$$u_a = u_a - \lambda_n F(u_a).$$

If we subtract this equation from (3.1) we have

$$u_{n+1} - u_a = (u_n - u_a) - \lambda_n (F(u_n) - F(u_a)).$$

Here we can apply the law of the mean and so obtain

$$u_{n+1} - u_a = (u_n - u_a)(1 - \lambda_n F'(\eta)), \tag{3.4}$$

where η is some value between u_n and u_a. From this, supposing $\lambda_n > 0$, $\lambda_n \to \lambda_a > 0$, we can draw the following conclusions. (These conclusions are called cases, although they are neither exhaustive nor mutually exclusive. The cases are illustrated by Figs. 1-4. In these figures the point on the curve where $u = u_k$ is labelled P_k).

Case 1. If throughout the interval from u_n to u_a we have

$$0 < \lambda_n F'(u) < 1 \tag{3.5}$$

then $u_{n+1} - u_a$ has the same sign as $u_n - u_a$ and is smaller. As long as the condition persists the sequence $(u_n - u_a)$ is monotone and decreasing in absolute value; if it persists indefinitely, as it will if (3.5) holds for $0 \leq u \leq 1$, then it must converge to a limit which is zero by the argument following (3.1). Hence in this case we get convergence, although it may be slow. Geometrically this case arises when the slope of the approximating line is greater than that of the tangent to the curve $x = F(u)$ throughout the interval. The situation is illustrated in Fig. 1. Case 1 will be called the **monotone case.**

Case 2. If throughout the same interval we have

$$1<\lambda_n F'(u) \tag{3.6}$$

then u_n and u_{n+1} are on opposite sides of u_a; as long as the condition persists the sequence $\left\{u_n - u_a\right\}$ is alternating. If further the condition

$$1<\lambda_n F'(u) < 2 \tag{3.7}$$

persists, the sequence is decreasing in absolute value; but we cannot say that it converges to zero. The conditions (3.6, 3.7) will both hold for all $n \geq k$ if they hold for $0 \leq u < 2$ and $0 \leq u_k \leq u_a < 1$. Geometrically Case 2 is that where the slope of the approximating line is less than that of the tangent; examples are Figs. 2a, 2b. Of course if $1 - \lambda_n F'(u)$ does not have a fixed sign we may have a combination of Cases 1 and 2; Fig. 3 is an example. Case 2 will be referred to as the **oscillating case.**

Case 3. Let $\left\{\theta_n\right\}$ $(n \geq 1)$ be a sequence of positive quantities less than 1 and $\left\{\zeta_n\right\}$ the sequence such that

$$\zeta_0 = 1, \quad \zeta_{n+1} = \theta_{n+1} \zeta_n.$$

Suppose that $\left\{\zeta_n\right\}$ converges to zero. Then a sufficient condition that the sequence $\left\{u_n - u_a\right\}$ converge to zero at least as rapdily as $\left\{\zeta_n\right\}$ is that

$$1-\theta_n \leq \lambda_n F'(u) \leq 1+\theta_n \tag{3.8}$$

throughout a sufficiently wide interval. If we suppose that throughout that interval

$$0<\alpha_n \leq F'(u) \leq \beta_n \tag{3.9}$$

(where the suffix n indicates that we are for the present allowing the interval to depend on n), then (3.8) will hold if

$$1 - \theta_n \leq \lambda_n \alpha_n \leq \lambda_n \beta_n \leq 1+\theta_n,$$

i.e. if

$$\frac{1-\theta_n}{\alpha_n} \leq \lambda_n \leq \frac{1+\theta_n}{\beta_n} \tag{3.10}$$

The best obtainable θ_n which satisfies (3.10) is that which makes the upper and lower bounds equal; this is

$$\theta_n = \frac{k_n - 1}{k_n + 1}, \text{where } k_n = \frac{\beta_n}{\alpha_n} \qquad (3.11)$$

3.6 These considerations show the reason for the extremely rapid convergence in the case of Newton's method. For under the assumption of convergence and continuity of F' (u) we have

$$\lim_{n \to \infty} \lambda_n F'(u) = 1,$$

when u varies in any way between u_n and u_a. Hence (3.8) holds for a sequence $\{\theta_n\}$ such that

$$\lim_{n \to \infty} \theta_n = 0$$

and consequently the sequence $\{\zeta_n\}$ converges faster than any geometric series. It is however possible to get divergence with Newton's method (see Fig. 4).

3.6a For the Eniac, however, extremely rapid convergence is not necessary. The calculation F(u) takes, as we shall see, approximately 60 addition times, while the interval between card-readings or printings is 0.6 second or 3000 add-times. Thus the bulk of the time, even for a relatively large number of approximations, is taken up by card feeding. Allowing for incidental calculations we may have as many as forty approximations without a great increase in the total time. A far more important consideration than speed of convergence is simplicity of programming.

3.7 Such simplicity can be attained by taking a fixed value for λ_n. Indeed we can take a value which is not only independent of n, but is fixed for a considerable range of values of a. To investigate this, let us suppose that we are concerned with a piece of the curve x = F(u) in which

$$0 < \alpha \leq F'(u) \leq \beta \qquad (3.12)$$

In the discussion of Case 3 above let $\theta_n = \theta$ be independent of n, and let $\alpha_n = \alpha$, $\beta_n = \beta$. Suppose that we are satisfied to terminate the iteration at the nth step is

$$\left| u_{n+1} - u_a \right| \leq 10^{-6}.$$

A sufficient condition that this will happen for $n \leq 40$ is $\theta^{40} \leq 10^{-6}$

i.e.

$$\theta \leq 0.708.$$

The maximum value of $k = \beta/\alpha$ is that satisfying (3.11), viz.

$$\frac{k-1}{k+1} = \theta,$$

i.e.

$$k = \frac{1+\theta}{1-\theta} = 5.85.$$

Thus if our computation is confined to a range for which the ratio of the maximum $F'(u)$ to the minimum is less than about 6, a constant value of λ can be found which will do for the whole computation. We can also take for u_0 any value such that $|u_a - u_0| \leq 1$; the simplest choice is to take $u_0=0$, but another choice will be recommended later on.

3.8 Since this restriction on $F'(u) = f'(u)$ is likely to be satisfied in a great many computations, the basic program of this report is planned for a fixed λ. Inasmuch as the computation is not sensitive to λ, the assumption that λ is a one figure number is justified. This arrangement greatly simplifies the programming; it is not necessary to use either the divider or, so far as multiplying by λ is concerned, the multiplier. The basic scheme may be modified to cover a wider interval in ways which the following discussion will suggest.

3.9 It is necessary, however, to incorporate some protection against divergence or excessively slow convergence in certain cases. This may arise through some error in planning, because we reach the boundary of an interval in which the above conditions hold, through some mal-functioning of the Eniac, through some roughness in the data, or what not. In such a case it is desirable to so program the Eniac as to sense the situation and make necessary adjustments or give a signal. In order to study methods of doing this we consider more in detail what happens in these abnormal cases. This will now be done under the assumption that

$$\lambda_n = \lambda > 0.$$

3.10 To this end we return to the hypothesis (3.6) of Case 2. From (3.1) we have

$$u_{n+k+1} = u_{n+k} - \lambda F(u_{n+k})$$

$$u_{n+1} = u_n - \lambda F(u_n).$$

Subtracting and applying the law of the mean we have

$$u_{n+k+1} - u_{n+1} = (u_{n+k} - u_n)(1 - \lambda F'(\eta)) \qquad (3.13)$$

where now η is between u_n and u_{n+k}. From this we can conclude that so long as we remain within an interval for which Case 2 holds, $u_{n+k+1} - u_{n+1}$ and $u_{n+k} - u_n$ have opposite signs. In particular, for $k = 2$, the differences $u_{2n+2} - u_{2n}$ all have one sign, while the differences $u_{2n+3} - u_{2n+1}$ all have precisely the opposite sign. Hence--always under the proviso that we remain within an interval for which Case 2 holds--the sequences $u_0, u_2, u_4, \ldots$ and $u_1, u_3, u_5, \ldots$ are both monotone, and in opposite directions. The case they are both monotone away from u_a we shall speak of as the **expanding case**; that where they are both monotone toward u_a the **contracting case**; it may happen that the sequences become stationary.

3.11 In the above we have distinguished the cases on the basis of behavior of $F'(u)$ within an interval. We may also make the distinction on the basis of the behavior of $F'(u)$ at the point or points $u = \eta$. Thus we can distinguish the cases thus

$0 < \lambda F'(\eta) < 1$	Monotone case
$\lambda F'(\eta) = 1$	Trivial ($u_n = u_a$)
$\lambda F'(\eta) > 1$	Oscillating case
$(1-\lambda F'(\eta_1))\,(1-\lambda F'(\eta_2)) < 1$	Contracting case
$(1-\lambda F'(\eta_1))\,(1-\lambda F'(\eta_2)) = 1$	Stationary case
$(1-\lambda F'(\eta_1))\,(1-\lambda F'(\eta_2)) > 1$	Expanding case

In the first three cases we take $k = 1$ in (3.13) and suppose η is between u_n and u_{n+1} (this is equivalent, so far as disntinguishing cases in an interval is concerned, to the earlier choice of η[1]).
In the last three cases η_1 is between u_n and u_{n+2}, η_2 between u_{n+1} and u_{n+3}. With this understanding we can pass from one case to another during the computation; the above interval conditions are sufficient conditions for the cases to occur or to persist.

3.12 If the contracting case persists indefinitely there will be two values u_b and u_c such that

$$\lim_{n\to\infty} u_{2n} = u_b \quad \lim_{n\to\infty} u_{2n+1} = u_c$$

This may also happen under other circumstances (for instance the stationary case.) If $u_b = u_c$, then we have convergence, and we have shown above that the limit must be u_a. If $u_b \neq u_c$, then since $u_o < u_a$ we have $u_b < u_a < u_c$. This case we shall speak of as **stabilized divergence.** There is some interest in investigating the conditions under which it occurs. From (3.1) we have

$$u_{2n+1} = u_{2n} - \lambda F(u_{2n})$$

$$u_{2n+2} = u_{2n+1} - \lambda F(u_{2n+1}).$$

Passing to the limit $n \to \infty$,

$$u_c = u_b - \lambda F(u_b)$$

$$u_b = u_c - \lambda F(u_c).$$

Subtracting the second equation from the first we have

$$2(u_c - u_b) = \lambda \left[F(u_c) - F(u_b) \right] .$$

[1] i.e. if the monotone or oscillating cases persist both the ratios $\dfrac{u_{n+2} - u_{n+1}}{u_{n+1} - u_n}$ and $\dfrac{u_{n+1} - u_a}{u_n - u_a}$ are necessarily positive in the one case and negative in the other. However this is not necessarily true at a time of transition from one case to the other.

Hence by the law of the mean, there is an η between u_b and u_c such that

$$\lambda F'(\eta) = 2.$$

This is a necessary condition.

3.13 One possibility remains to be considered. It is conceivable that at some stage the u_{n+1} calculated by (3.1) may be outside the region in which the hypotheses of section 1 hold. In such a case almost anything can happen; the process may even converge to another root of f(u) (which, under the special assumptions, is a cubic). This possibility will be known hereunder as **Case 4.**

3.14 We may now consider what may be done in programming this procedure for the Eniac to guard against these abnormal occurrences. It is supposed, of course, that at the end of each approximation the Eniac will test to determine whether F = 0 to within the tolerance, and according to the result of this test, proceed to the next step or discontinue the iteration. At the same time the Eniac can make other discriminations to test for the abnormal cases, and give a signal or take appropriate actions when they occur. Methods of doing this will be considered later. We consider here what some of these auxiliary tests may be.

3.15 If we include a test to determine whether u is within the interval from 0 to 1, or, more generally, 0 to h, we shall have protection against indefinite continuance of the expanding case and also some protection against Case 4. We shall not, however, have absolute protection against Case 4. For if, due to some error, our hypotheses are not fulfilled, we may have Case 4 even in the interval from 0 to 1. Moreover our assumptions pertain to the nature of the function f(u); and even though the physical function x(t) may satisfy the hypotheses, it is conceivable f'(u) may be zero, particularly if there is roughness in x(t)[1]. This possibility would require too detailed an investigation to be given here. The difficulty in that case pertains to the adequacy of f(u) as an approximation and not to the method of inversion. In the nature of things it is not possible to have absolute protection against Case 4.

3.16 We may also count the number of approximations in a stepper, changing the programming when a certain number is exceeded. With this test and the preceding one we have, except for the remote possibility of an anomalous Case 4, protection against all abnormal cases.[2] If these abnormalities are expected to occur rather rarely, so that all that is needed is to stop the machine and signal the operator in case of abnormality, then these protections suffice.

3.17 On the other hand there are many stiuations where it is desirable to distinguish between different causes of abnormality. Thus the number of approximations will be excessive in the monotone case when the value of λ is too small, in the oscillatory case when the value of λ is too large. The above procedures do not distinguish between excessively slow monotone convergence and stabilized divergence.

3.18 These cases can be distinguished by having the Eniac, in testing the vanishing of F, take a different action in case $F > 0$ from what it does if $F < 0$. According to our assumptions we start with $u_o < u_a$, $F < 0$. As long as we have the monotone case we shall have $F < 0$; as soon as we have $F > 0$ we shall know that the oscillating case has occurred. We can even go further than this and allow the Eniac to take

[1] Thus in Fig. 5 we have an example of a cubic through the four points with evenly spaced abscissas and increasing ordinates, which nevertheless has a maximum.

[2] This ignores the effect of λ on the tolerance (see par. 3.21).

remedial measures automatically. Thus, if we have a series of values of λ available, the Eniac can insist on the monotone case by decreasing the value of λ as soon as $F(u) > 0$ occurs. Then, if the number of steps is excessive it must be because λ is too small, and we can program an increase.

3.19 Let us discuss this situation theoretically under the assumption that

$$F''(u) < 0,$$

an assumption which is verified in most firing table computations. Then $F'(u)$ is a decreasing function; the curve is concave downward. The maximum $F'(u)$ is at the beginning of the computation; if λ is such that we have the monotone case at the start, we shall have it throughout. If ϵ is the tolerance on $u_{n+1} - u_n$ (which is λ times the tolerance on $F(u_n)$), and if $\theta^n = \epsilon$, then the condition for monotone convergence within n or less approximations is (3.10) with $1 + \theta$ on the right replaced by 1[1])

$$\frac{1-\theta}{\alpha} \leq \lambda \leq \frac{1}{\beta} \quad . \tag{3.14}$$

The right inequality, which is sufficient for the monotone case, is satisfied if $\lambda = \frac{1}{\beta}$. Suppose now that $u_{n+1} - u_n$ has failed to be within the tolerance. Let

$$\rho_n = \frac{u_{n+1} - u_n}{u_n - u_{n-1}} = 1 - \lambda F'(\eta_n)$$

where now

$$u_{n-1} < \eta_n < u_n.$$

Then ρ_n is increasing with n. Hence

$$\epsilon = \theta^n < u_{n+1} - u_n < \rho^n{}_n \ (u_1 - u_0) \leq \rho^n{}_n.$$

Therefore for all $u \geq u_n$ we shall certainly have

$$1 > 1 - \lambda F'(u) > \rho_n > \theta. \tag{3.15}$$

Now let

$$\theta = 1 - \frac{1}{r}, \qquad r > 1.$$

Then (3.15) gives us, on multiplying through by r

$$r > r - r\lambda F'(u) > r - 1,$$

i.e.

$$0 < r\lambda F'(u) < 1.$$

[1] In the monotone case we have 1 instead of $1+\theta_n$ on the extreme right in (3.8), and hence in (3.10).

Thus if $r\lambda$ is substituted for λ, we shall still have the monotone case. This may be repeated; if so each set of steps until a change in λ will be known as a grade. If we have p grades with the same r the maximum value of $\frac{\beta}{\alpha}$ for which we shall get convergence in not more than pn steps, is r^p. The values of n and r^p are shown for various values of r and p in the following table. It is noteworthy that in no case where pn $\leq$ 40 do we get a larger value of $\frac{\beta}{\alpha}$ than in the above discussion of Case 3; this is because in insisting on the monotone case we cut down $\frac{\beta}{\alpha}$ by a factor of $\frac{1}{1+\theta}$. The last line of the table shows the value of $\frac{\beta}{\alpha}$ for one stage of non-monotone convergence.

Calculation of θ, n, r such that

$\theta = (1 - \frac{1}{r})$ $\qquad \theta^n = 10^{-6}$

r	1.5	2	3	4	5
θ	0.333333	.500000	.666667	.750000	.800000
n	13.0	19.9	34.1	48.0	61.9
r	1.5	2	3	4	5
r^2	2.25	4	9	16	25
r^3	3.375	8	27	64	125
r^4	5.0625	16	81	256	625
r^5	7.59375	32	243	1024	3125
r(1+θ)	2	3	5	7	9

$r = \frac{1}{1-\theta}$

n	10	12	15	20	25	30	40	50	60	80	100
θ	.251	.316	.398	.501	.575	.631	.708	.759	.794	.841	.871
r	1.34	1.46	1.66	2.00	2.36	2.71	3.42	4.14	4.86	6.31	7.75
r^2	1.78	2.14	2.76	4.02	5.55	7.34	11.7	17.2	23.6	39.8	60.1
r^3	2.38	3.13	4.59	8.06	13.1	19.9	40.2	71.1	115	251	466
r^4	3.18	4.57	7.62	16.2	30.8	53.9	138	294	559	1581	3611
r^5	4.25	6.69	12.7	32.4	72.5	146	471	1220	2715	9965	27992
r(1+θ)	1.67	1.92	2.32	3.01	3.71	4.42	5.85	7.28	8.72	11.6	14.5

3.20 In the foregoing we have been considering a series of approximations for a single round. This is not trivial; indeed if we are approaching a maximum of F(u)--as in certain firing table calculations near maximum range--we may have large values of $\frac{\beta}{\alpha}$ even in a small interval. But the discussion takes on a broader significance for a group of rounds with a large number of different values of a. Of course every time we read a new card we change f(u); but the assemblage of functions f(u) forms an approximation to x(t) which we may reasonably suppose becomes less and less steep as we proceed. If now we take $u_o = 0$ whenever we read a new card, while in the case of two or more rounds without a change of input card we let the u_o for one round be the final value for the preceding round, then the assumptions made for a single round will apply to the group as a whole. In such a case with several values of λ attached to the various positions of a stepper we can obtain as adequate a coverage of a large variation in F'(u) as if we calculated a separate λ for each value of a. Of course for a large n we may lose a card cycle or so when the stepper is stepped; but since this occurs only a few times in the entire group, the time lost is insignificant.

3.21 There is, however, one serious difficulty in all this. The above discussion has reference to a fixed tolerance on $u_{n+1} - u_n$; whereas what we want is a determination of u_a with a prescribed error. Now we have

$$\begin{aligned} u_{n+1} - u_n &= - \lambda F(u_n) \\ &= \lambda \left[F(u_a) - F(u_n) \right] \\ &= \lambda (u_a - u_n) F'(\eta) \end{aligned} \qquad (3.16)$$

where $u_n < \eta < u_a$. Hence

$$u_{n+1} - u_n \geq \lambda \alpha (u_a - u_n)$$

so that if

$$\left| u_{n+1} = u_n \right| < \epsilon$$

we have

$$0 < u_a - u_n < \frac{\epsilon}{\lambda \alpha} = \epsilon \frac{\beta}{\alpha}. \qquad (3.17)$$

This shows that slow convergence is objectionable not only because of the time element but because of its effect on the accuracy of the result. If α is a known positive number, (3.17) shows how ϵ can be determined to insure against excessive error; in the case of a multi-grade procedure, naturally the smallest value of λ must be used in order to be safe. If the α is very small, or if, as in the case where we are approaching a maximum, it is unknown (note that an α always exists if a u_a exists, viz. $\alpha = F'(u_a)$--in such a case our hypotheses are fulfilled for $0 \leq u \leq u_a$,) it is advisable to take into account the effect of the second derivative. Let m be a lower bound for -F"(u). Then instead of (3.16) we have

$$u_{n+1} - u_n = - \lambda \left[F(u_a) + (u_n - u_a) F'(u_a) + \frac{(u_n - u_a)^2}{2!} F''(\eta) \right]$$

where η is within the same limits as before. Then

$$\epsilon \geq u_{n+1} - u_n \geq \lambda \alpha (u_a - u_n) + \lambda \frac{m}{2} (u_a - u_n)^2.$$

This gives (solving the inequality by completing the square and rationalizing the numerator)

$$u_a - u_n \leq \frac{2\epsilon}{\lambda\alpha + \sqrt{2\lambda m\epsilon + \lambda^2\alpha^2}} . \qquad (3.18)$$

If we put $\alpha = 0$, this gives as upper bound on the error near a maximum

$$u_a - u_n \leq \sqrt{\frac{2\epsilon}{\lambda m}} = \sqrt{\frac{2\beta\epsilon}{m}} . \qquad (3.19)$$

3.22 Let us pause a moment to consider some modifications. If we are to use a stepwise variation in λ, why not determine λ advantageously at the beginning of each round? A suitable value of β, call it $\bar{\beta}$, is available in the machine: viz. $\bar{\beta} = F(0) - F(-1) = x_k - x_{k-1}$--in special cases we may also have $\bar{\beta} = F'(0)$. To calculate $1/\bar{\beta}$ in such case would require the divider, and also further use of the multiplier to form $\lambda F(u)$. This procedure, although it may well be feasible if the number of secondary functions is reduced, is impossible in the basic scheme. An alternative is to use a stepwise variation of λ as above, with a discrimination at the beginning of every round, so as to insure that

$$\frac{1}{r} < \lambda\bar{\beta} < 1.$$

This test makes it certain that the round will start with the most advantageous value of λ. That decreases the probability of excessively slow convergence, but will not altogether eliminate it, as the above discussion shows; however it is then rather unlikely that the lost time or the error will be serious.

3.23 Of course a similar discussion can be made if

$$F''(u) > 0.$$

The situation is, in a sense, converse to that above. We shall have the minimum $F'(u)$ at the beginning, if we set $\lambda = \frac{1-\theta}{\alpha}$ we are protected against excessively slow convergence throughout the calculation. If at the **m**'th approximation our tests show that oscillation has occurred, then for all $u > u_m$

$$\lambda F'(u) > 1,$$

hence $\frac{\lambda}{r} F'(u) > \frac{\lambda}{r} = 1-\theta.$

We can therefore decrease λ by a factor r and still be protected against excessively slow convergence. With appropriate modifications we can then have a multi-grade computation with the same increases in β/α as before. The procedure will be slightly faster than that for $F''(u) < 0$, since the change in λ may occur for a small value of **m**. We also have automatic protection against error due to too small a λ.

3.24 On the other hand there is the difficulty that when the tests show λ should be decreased, the value of u_n is already incorrect; in order to have a theoretically correct procedure it is necessary to retain u_{n-1} somewhere. Of course we might go back to $u = 0$, or continue with the same λ to u_{n+1}, which the theory shows is $\leq u_a$; but although, in view of our safeguards, one of these procedures might work out as a practical matter, the example of Fig. 6 shows difficulties which are theoretically possible. The procedure is also not so well adapted for operating near a maximum or minimum of **x**(t).

3.25 Let us now sum up this discussion. Between approximations the Eniac can make discriminations (a), (b), (c), with alternatives as follows

(a1)	$\left\|\lambda F(u_n)\right\| < \epsilon$
(a2)	$\lambda F(u_n) < -\epsilon$
(a3)	$\lambda F(u_n) > \epsilon$
(b1)	$0 \leq u_{n+1} < h$
(b2)	$u_{n+1} < 0$
(b3)	$u_{n+1} > h$
(c1)	$n \leq N$
(c2)	$n > N$

where h is an upper limit on u. These discriminations can be made in any order; having made one discrimination we proceed to the next in case the first alternative is the one for which a new approximation is desired. In the schemes below it is convenient to make the tests in the order (c) (a) (b). It will be noted that the essential discrimination is (a); the discriminations (b) and (c) are only safeguards which could be omitted if such safeguards were unnecessary.

3.26 In the basic scheme below all three discriminations (a) (b) (c) are provided for. There is a single value of λ. The alternatives (a3), (b2), (b3), (c2) each have an open output; if one of these alternatives occurs the Eniac will stop, the positions of the steppers indicating the reason--such behavior will be called an **error signal.** The case (a1) signals that the iteration is complete and orders the secondary interpolations to begin; while (a2) orders a new approximation through (b). The limit h is set at 2. This scheme is conservative. Not only does it contain the tests (b) and (c), but it insists on the monotone case and so restricts $\frac{\beta}{\alpha}$ unnecessarily. In fact the basic scheme is not intended to be used as such; it is designed, as its name suggests, as a basis which can be modified according to circumstances. Aside from the possible omission of tests (b) and (c) the following modifications are suggested by the foregoing theory. In this discussion the stepper used in (c) will be called the **counting stepper.** Further modifications will be considered in sections 8, 9, and 10.

(A) In case we wish to use only one λ but do not wish to insist on the monotone case, we should connect (a3) to the same output as (a2). In this case it is essential because of the possibility of stabilized divergence, that (c2) be an error signal.

(B) If we wish to incorporate the method outlined above for having a stepwise variation in λ controlled by a counting stepper for $F''(u) < 0$, the counting stepper should have several positions. Each position will have a counter switch setting, and the stepper will step automatically when the number set on the switches--which need not be the same number in the various positions--is exceeded. The output of each position, except the last, will be the same as (c1) except for the difference in the values of λ; the last position should give an error signal (to prevent the stepper from cycling back to the initial position.) It is essential that (a3) give an error signal. For further discussion of this modification see section 8.

(C) If we wish to use the step-wise scheme for $F''(u) > 0$, the output of (a2) should order a new approximation through a stepper with a different value of λ for each position. The output of (a3) should step this stepper to a new position, replace u_n by u_{n-1}, if available, and then do the same as (a2). If reliance is placed on safeguards to warn of pathological conditions, we may replace u_n by 0 or, using the same u_n, go to the same output as (a2) before stepping the stepper. The output of (c2), (b2), (b3) should be an error signal.

(D) Since a stepper can be stepped in the negative direction by stepping it one stage less than a complete cycle in the forward direction, the modifications (B) and (C) can be combined. This modification is not tied to any assumption as to $F''(u)$. On the other hand some of the protection against a Case 4 error has been removed, and without an additional counting stepper it is conceivable there might be indefinite oscillation between a larger and a smaller value of λ. The modification is also more difficult to program.

We also have the following, which are not modifications in III, but involve changes in II;

(E) In case we wish to use a stepwise variation in λ controlled at the beginning of a round we require a discrimination in process II. Consider the case where $F''(u) < 0$. Suppose, as a first assumption that we have $\lambda\beta \leq 1$ taken care of. Then we require a two-way discrimination on the relative magnitudes of $r\lambda\beta$ and 1; if $r\lambda\beta > 1$ we proceed with the calculation, if not we step to the next value of λ and retest. The value of λ is of course a program using shifters and repeat switches; we have to put r through this program while later in III we need to put $F(u_n)$ through it; this requires a complication in the programming. The complication is even greater if (E) is combined with (B). The matter will be discussed further in section 8.

(F) A similar modification can be made if $F''(u) > 0$. This will not be further discussed.

(G) If we wish to form a λ for each round by division, we require the divider or an equivalent procedure involving steppers etc. The use of the divider is only feasible if there are less than three secondary functions. In that case the modification is obvious. Except for a note in section 8, this modification is not further considered in this report.

3.27 No attempt will be made to program all of these modifications. Methods of adding certain of them to the basic scheme, and also certain further modifications will be considered in sections 8 and 9.

3.28 If a counting stepper is used there is a difficulty with regard to clearing it. For modifications (B) and (D) it is necessary that the stepper counter be cleared, without disturbing the stepper position, at the end of every round (in (D) at the end of every grade); while the stepper itself is cleared at the end of every group. For the other modifications and the basic scheme the counter stepper will need to be cleared at the end of every round. As the Eniac is at present designed there is no direct provision for doing this. The stepper decade counters can be cleared only by either the initial clear gate or the counter reaching the value set on the switches. This is a defect in the design of the Eniac as the engineers on the project have themselves admitted. With the addition of one or two tubes per decade of the master programmer, it would be possible to arrange it so that either the clearing of the counter can be programmed or the clearing will take place automatically whenever the stepper is stepped or cleared. In the latter case clearing of the counter without stepping can be achieved by stepping the stepper through a complete cycle.

4. METHODS OF DISCRIMINATING

4.1 It is evident from the above discussion that in the program of inverse interpolation it is frequently necessary for the Eniac to take different actions according to the relative magnitudes of two quantities or, what amounts to the same thing, according to the sign of their difference. Such action by a computing machine is known as **discrimination.** We consider here ways of making the various discriminations needed in the present problem expeditiously.

4.2 The fundamental principle of discrimination in the Eniac is that the PM of an accumulator gives a means of detecting the sign of its contents. When the accumulator transmits from one of its digit outlets, the PM of that outlet transmits 9 pulses or no pulses according to circumstances. If now a line is taken from that PM to a program input (in an accumulator or stepper) the flip-flop will be set in the case of 9 pulses and will not be set in the case of no pulses. At the end of the addition time[1] we shall get a program output in the case of 9 pulses and none otherwise. If we want an output in neither case, but different outputs in the two cases, the above program output can be taken to the direct input of a stepper; then the next regular input to that stepper will come out in a different position in each case. In the following discussion it is assumed this is done[2].

4.3 From the above it would appear that a digit outlet of an accumulator and a stepper are necessary for each discrimination. Considering the number of discriminations to be made in this problem, considerable equipment would be used up on that basis. It is therefore worth while to discuss ways in which equipment can be conserved. In this discussion it will be convenient to use letters x, y, z, u, a, b, etc. to denote variables which are not necessarily the same as those so denoted in the body of the report. The accumulator in which the quantity is tested will be called the **test accumulator,** the outlet from which the PM is taken will be called the **test outlet,** and the quantity tested in the test accumulator will be called the **test quantity.** The case where there is a signal from the PM will be called the **positive case;** the other the **negative case.**

4.4 Suppose that after testing a quantity x in the test accumulator we put in a second quantity y and test again. Then the position of the stepper will indicate the number of positive tests. Evidently if we have a

[1] This is the same addition time as that in which the nine pulses are emitted. The program output will be simultaneous with that which would ordinarily follow immediately after the transmission. In the diagrams the input to the above dummy program comes from the PM of a transmission occurring in the same addition time, not the preceding.

[2] i.e. we are ignoring other methods of doing this such as taking lines from the PMs of both the A and the S outlets. The latter procedure ties up two accumulator outlets; in the present problem steppers are less in demand than accumulator outlets.

6-position stepper we can test as many as five quantities and get a discrimination on the number of positive cases. A test of this character will be called a **cardinal test,** because it tests a cardinal number.

4.5 An example where a cardinal test is useful is where we have $m \leq 5$ given quantities $x_1, \ldots x_m$, concerning which it is known that $x_1 < x_2 < \ldots < x_m$, and we wish to determine in which of the m+1 half open intervals into which $x_1, \ldots x_m$ divide the real line another quantity y lies. This can be done by a cardinal test on the m quantities $y-x$, $i=1,2, \ldots, m$, and requires an m+1 position stepper.

4.6 Another type of multiple test is the following. Suppose we make a test T_1; a second test T_2 if and only if T_1 is positive; a third test T_3 if and only if T_2 is positive; and so on, the negative cases and the last positive case giving the different alternatives. With a six position stepper we can make as many as three such tests giving four alternatives. This can be done by having a regular input to the stepper after each test, and having the stepper stepped on the outputs of the second and fourth positions. The first, third, fifth, and sixth positions give the four alternatives. Such a multiple test, involving two or three single tests will be called a **chain test.**

4.7 In general if a given test leaves two or more positions of a stepper free, those positions may be used for a further test provided the stepper is first stepped into the first of the free positions.

4.8 Let us now consider more in detail methods of making certain tests. The question of what happens when the test quantity is zero, which was ignored in the above discussion, will now be taken into account. This behavior is slightly different in the different ways of making a test.

4.9 If we are testing a single quantity x, then the A outlet will give a positive test if $x < 0$; the S outlet will give a positive test if $x \geq 0$.

4.10 If we are making a simple comparison between x and z there are 8 ways of making the test as shown in Table 4.1 below. Here the column headed t gives the test quantity and the columns headed A and S the conditions for a positive test on the A and S outlets respectively. In the cases c and d the 1 indicates a unit in the right hand-most decade; such cases can arise through the addition or deletion of a subtract correction pulse.

Table 4.1

	t	A	S
a	$x-z$	$x < z$	$x \geq z$
b	$z-x$	$x > z$	$x \leq z$
c	$x-z-1$	$x \leq z$	$x > z$
d	$z-x-1$	$x \geq z$	$x < z$

Note that the cases b and c separate the alternatives $x \leq z$ and $x > z$; the others the alternatives $x < z$ and $x \geq z$.

4.11 Suppose now we have three quantities x, y, z concerning which we wish to distinguish the cases:

Case 1	$x \leq z$	$y \leq z$
Case 2	$x \leq z < y$	
Case 3	$x > z$	$y > z$
Case 4		$y \leq z < x$

Then we are making tests of the type b or c (in Table 4.1) on (x, z) and (y, z). For the four possible ways of combining the tests of Table 4.1, the positive tests are indicated in Table 4.2. Here the first symbol in any pair refers to the (x, z) test, the second to the (y, z) test.

Table 4.2

	Case 1	Case 2	Case 3	Case 4
bb	SS	SA	AA	AS
bc	SA	SS	AS	AA
cb	AS	AA	SA	SS
cc	AA	AS	SS	SA

4.12 In many applications of this test we know that $x < y$ so that Case 4 is excluded. Then the possibilities bb and cc are suitable for a cardinal test to distinguish the remaining three cases. In other instances Case 4 may be more or less pathological and need to be guarded against in some way. In such an event we might use one of the other types of test, or a chain test.

4.13 An example of this kind occurs in Process I where we seek to test the inequality (in the notation of par. 2.3)

$$x_k \leq a < x_{k+1}$$

Here for **x, y, z** we have x_k, x_{k+1}, a, respectively. In this example Case I would require the reading of a new card; Case 2 would be the signal to proceed with the primary interpolation; while Cases 3 and 4 would not normally arise. Hence the natural procedure would be to line up Cases 3 and 4 in a position which gives an error signal. This is done in Fig. 9a below by a chain test of type bb on the S outlet. This leaves two positions of the stepper free and brings Case 1 out of the fourth position. The last two positions can, therefore, be conveniently used to prevent the reading of cards past a maximum of x(t). Two such amplified tests are shown in Figs. 9b and 9c (in Fig. 9b we think of z as $\dot{x}$); they will be discussed in section 8 below.

4.14 Let us now suppose that the cases to be distinguished are

Case 1	$x \leq z$	$y < z$
Case 2	$x \leq z \leq y$	
Case 3	$x > z$	$y \geq z$
Case 4	$x > z \quad y < z$	

Here the test on (x,z) is again of Type b or c, while that on (y,z) is of type a or d (in Table 4.1). Instead of Table 4.2 we have the analogous Table 4.3.

Table 4.3

	Case 1	Case 2	Case 3	Case 4
ba	SA	SS	AS	AA
bd	SS	SA	AA	AS
ca	AA	AS	SS	SA
cd	AS	AA	SA	SS

4.15 If $x \leq y$ then Case 4 is impossible. To make the distinction between the other three cases by a cardinal test we must use Types bd or ca. Since type c and d of Table 4.1 are slightly more difficult to program than Types a and b, the tests of Table 4.2 are preferable when $x < y$. But when $x=y$ Table 4.3 must be used. In that event a discrimination by Table 4.3 will be called an equality discrimination. Equality discriminations thus form the principal field of application of Table 4.3. They are evidently symmetric in x and z. Since three positions of the stepper are used, two such tests can be made with a single test outlet and one six-position stepper.

4.16 An instance of this test which is not an equality discrimination is what is known later as the auxiliary test. In this test the three alternatives are

$$(1) \quad w_k + 2z_k < 0$$

$$(2) \quad w_k - 2z_k \leq 0 \leq w_k + 2z_k$$

$$(3) \quad 0 < w_k - 2z_k$$

Of these alternatives the second is regarded as normal and allows the computation to proceed; while the first and third signal card reading. Under the hypothesis that $z > 0$ Case 4 is impossible, and the effect of the test is confine computation to the case where w/z is between -2 and +2. This test has naturally nothing to do with the general theory of the basic scheme; it is included as an example of, and to save room for, special tests which will vary with the particular application. Its motivation is a possible application to antiaircraft firing tables where z is specialized to $\dot{x}$ and w to $\dot{y}$; then w/z is the slope of the trajectory which must then lie between -2 and +2. The test is programmed as a chain test of type ba on the S-outlet in Fig. 8. Since in this test the limiting cases of equality are of little importance an essentially equivalent test of type bb (Table 4.2) could have been programmed; but such a program, when tried, appeared to use the equipment less efficiently than that in Fig. 8.

4.17 An instance of an equality discrimination is the test for starting a new group. We must have the out-argument b in one of the printer accumulators. After positioning a card we read and compare its φ with b. If they agree the card belongs to the correct group and the computation proceeds. If $\varphi > b$ the card belongs to the next group, and the adjustments to start a new group are made. If $\varphi < b$ it may mean an error has occurred, in which case an error signal is appropriate; or it may mean that the rest of the group is to be skipped, in which case the reading of cards is in order.

4.18 This test is combined, in the program below, with another equality discrimination. The latter, which is purely a safety precaution, is a comparison of a card number c, supposed to occur on all the input cards, with a similar number c* stored in the machine. An error signal is given in either of the cases of inequality. For the combined programming of these two tests see the two alternative schemes in Figs. 7a, 7b. Of these the former has been selected for the basic program.[1] Note that in either case the -1 needed for tests of type c is obtained by adding a subtract correction pulse to the negative of the test quantity before subtracting it into the test accumulator.

[1] The advantage of Fig. 7a is that it saves a dummy at the expense of an addition time. If the saving of time should be more important Fig. 7b could be used.

4.19 Let us now consider the discrimination between the approximations as described in par. 3.25. It is convenient to go through the counting stepper first and then form $\lambda F(u_n)$, so that the discriminations (a) (of par. 3.25) impose a tolerance of $u_{n+1}-u_n$ rather than on $F(u_n)$. This preliminary adjustment, which is of no interest here, is shown in Fig. 18. To make the discrimination (a), assume that the tolerance ϵ can be imposed on λF by cutting off some of the last digits and requiring the so amputated λF, here called $\lambda \bar{F}$, to be zero. Then the discrimination (a) is an equality discrimination with $z= \lambda \bar{F}$, $x=y=0$. If $-\lambda \bar{F}$ is carried into the test accumulator by a subtraction, then the deletion of the last few digits also deletes the subtract pulse, thus furnishing the -1 required by the above theory. Since the test requires only three positions of the stepper, and since the discrimination (b) can be made by a cardinal test of the type which uses Table 4.2, the two can be combined on a single stepper and test outlet. A program on this basis is shown in Fig. 19.

4.20 In all these tests the stepper should be one without attached decades. To insure that the stepper is cleared at the beginning of the test, the stepper direct clear input should be pulsed at the start. This would, if it were desirable, allow the test outlet to be used for other purposes while the test is not in progress.

5. THEORETICAL CONSIDERATIONS RELATED TO THE FORMATION OF LINEAR EXPRESSIONS

5.1 The problem of forming differences differs from that in a previous report[1] in that instead of reading the functional values from a function table, we have these values in accumulators. In fact three of these values are stored in accumulators by virtue of our previous hypotheses; while the fourth, which is in the constant transmitter, is put into an accumulator in order to combine it with the others. The formation of differences is thus a special case of a general theorem, which, since it is likely to be useful in other connections, will be stated and proved here. The theorem gives a general method of transforming a set of n quantities $x_1, x_2, \ldots, x_n$ into another set $y_1, y_2, \ldots, y_n$ by means of n accumulators. In this section the x_i, y_i, have no connection with those in the rest of the report. This theorem is as follows:

5.2 Theorem Let the n accumulators $A_1, A_2, \ldots, A_n$ contain initially the n quantities $x_1, \ldots, x_n$ respectively. Then a necessary and sufficient condition, that $y_1, \ldots, y_n$ be such that there exist a series of transfers between the $A_1, \ldots, A_n$ without clearing which ends with $y_1, \ldots, y_n$ in $A_1, \ldots, A_n$ respectively, is that there exist a matrix of integers $C = (C_{ij})$ such that

$$y_i = C_{i1}\ x_1 + C_{i2}x_2 + \ldots + C_{in}x_n \quad i = 1, 2, \ldots, n \text{ and } \left| C_{ij} \right| = 1$$

where $|C_{ij}|$ is the determinant of the matrix C.

5.3 Proof of Necessity. If the y's can be generated as stated in the theorem, then they can be generated by a series of steps each of which consists of transmitting once the contents of some one of the A's to another

[1] BRL Report #613, **A Study of Fourth Order Interpolation on the Eniac** by Haskell B. Curry and Max Lotkin.

oen of the A's; for multiple transfers can be accomplished by a succession of such single transfers. Suppose now that after m such steps we have a set of y's which satisfy the stated conditions with a matrix C. Let the (m+1)st step consist of adding A_i to A_k or subtracting A_i from A_k. Then the new set of y's will satisfy (1) with a matrix C' formed by adding the ith row to the kth row (or the corresponding subtraction). The matrix C' will have integer elements if C does; moreover C and C' have the same determinant; hence the condition holds after m+1 steps. Since for m=0 the matrix C is the unit matrix

$$C_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$$

the necessity follows by induction on m.

5.4 Proof of Sufficiency. It will suffice to show that by succession of steps, each consisting of modification of one row by the addition to it (or subtraction from it) of the elements of another row, the matrix C can be reduced to the unit matrix; for then the reverse process, when interpreted in terms of accumulators, gives a method of generating $y_1, \ldots, y_n$. This reduction, which is well known, will be presented here for completeness. It will be divided into three stages: 1) the reduction to the form where all the elements in the last column, except the last, are zero, while the last is 1; 2) reduction to the semidiagonal form where all the elements in the principal diagonal are 1 and all the elements above the principal diagonal are zero; 3) the reduction of such a semi-diagonal matrix to the unit matrix.

5.5 To complete the first stage: the elements in the last column are not all zero since the determinant is not. Choose an element whose absolute value is positive and as small as possible; let this be C_{in}. By adding this to or subtracting this from each of the other rows a sufficient number of times we can bring it about that the absolute value of each of its other elements in the last column is less than $|C_{in}|$. If not all these other elements are zero we can choose another i and repeat the process. Eventually we arrive at a case where $C_{in} \neq 0$ while all other elements in the nth column vanish. Then C_{in} divides the determinant and so $C_{in} = \pm 1$. If it is -1 subtract the i'th row from the j'th and then add the j'th back to the i'th; we then have $C_{in} = 1$ while all other elements in the n'th column vanish. Finally if $j \neq n$ add the j'th row to the n'th and then subtract the n'th from the j'th.

5.6 To complete the second stage: suppose that we have reduced the matrix to a form in which the k last columns have all zeros above the main diagonal and all 1's in it. Then we can apply the operations of the first stage to the first n-k rows and columns; if the corresponding operations are carried out on the whole matrix the last k rows and the last k columns will not be affected. In this way we can increase the number k by one unit. So we can continue until k = n. (Note that the nth step is vacuous; for when k = n-1 we have $C_{11} = |C_{ij}| = 1$.)

5.7 To complete the third stage it is merely necessary to add the first row to (or subtract it from) each of the others until all the elements in the first column below the main diagonal are zero; then add the second row to (or subtract it from) each of the succeeding ones until all the elements in the second column below the main diagonal are zero, and so on.

5.8 This theorem can be paraphrased by saying that the most general transformation which can be accomplished by non-clearing accumulators alone is a modular transformation. Since the modular transformations form a group, such a transformation can be inverted; this is also clear directly from the fact that the process above outlined is reversible.

5.9 Note that the above process of reduction is much simpler, and is also essentially unique, in case the matrix C is semi-diagonal to begin with. This is the case if the y's are the differences of the x's; it is also apt to be the case if the y's are the coefficients in an interpolation formula of any kind. The program for forming the differences in the present problem which this theorem suggests is shown in Fig. 27. On this same figure the formation of differences what is essentially the difference method of the report mentioned in par. 5.1 is shown for comparison.

6. SCHEDULE OF STAGES FOR THE BASIC SCHEME

6.1 We have now discussed theoretically how various partial computations which are essential for inverse interpolation may be performed. We must now turn to the problem of making a program for the whole computation in accordance with these principles.

6.2 According to par. 2.8 the computation is broken into pieces, called stages, of such a character that the stages are repeated as wholes in the different parts of the computation. This division can, in principle, be made a priori on the basis of the foregoing discussion. In practice a preliminary schedule of stages was made, and revised as the study went on in the light of experience with the detailed programming. The schedules given below give the final revision.

6.3 Before proceeding to the schedules an illustration will be given of one of the principles according to which division into stages was made. Consider the operation of constructing f(u) when u is given. This is done with a certain set of coefficients A, B, C, D, in each step of the primary interpolation, and with different coefficients in each of the secondary interpolations. Now the processes of multiplication and the intermediate adjustments are the same, aside from the differences in the coefficients, in all of these instances. These processes can be separated from the generation of the coefficients as a connected sequence of programs. This is known hereafter as Stage III 2. This stage is repeated each time an f(u) (or the analogous expression in the secondary interpolations) is calculated. It opens gates for the reception of coefficients at appropriate times, but the coefficients themselves must come from another stage. The latter stage is III 1 in the case of the primary interpolation and IV 2, V 2, and VI 2 for the respective secondary interpolations. Thus the program elements are grouped into stages according to the frequency, so to speak, at which they repeat; when an operation involves elements which recur more frequently than the others, the more frequently recurring elements can be grouped into a stage by themselves, which stage can run concurrently with other stages. Of course a further division into stages may be made for other reasons.

6.4 We can now proceed to the schedule of stages. Each stage is given an arabic numeral which is used in connection with the number of the process where the stage first occurs (it may be repeated in later processes.) After the number and name of the stage and a description of its function thê input (i) and the outputs (o1, o2, . . .) are listed. The schedule may be studied in connection with the overall program in Figs. 28-31, which will be explained in par. 7.8, and the flow chart in Fig. 44. Note that the start of the computation is through stage I 10.

I. SET UP.

1. Card reading. Pulse R_i to order reading of new card. The reader interlock (R_l) is connected on same line as R_i as there is no use for the interlock feature. (See discussion of I 5.) May be concurrent with part of I 8.

i:	I 5 (o1), I 9 (o2)
o1:	I 2

2. Group test. Test on φ and c. Test $c = c^*$, $\varphi = b$ as described in par. 4.17f. Read φ and c from card, also c* from CT. May be concurrent, in part, with I 8.

i:	I 1, I 8 or I 7 (o2)
o1:	$(c = c^*, \varphi = b)$ I 3
o2:	$(c = c^*, \varphi > b)$ I 7
o3:	$(c = c^*, \varphi < b)$ Error or I 9[1] (See par. 8.6)
o4:	$(c > c^*)$ Error
o5:	$(c < c^*)$ Error

3. Auxiliary test. This is described in par. 4.16. As stated there the test is only an example. It provides a place for insertion of tests which will presumably vary with the problem.

i:	I 2 (o1)
o1:	$(w_k - 2z_k \leq 0 < w_k + 2z_k)$ I 4
o2:	$(0 < w_k - 2z_k)$ I 5
o3:	$(w_k + 2z_k < 0)$ I 5 (or I 7)[2]

4. Principal test. Test whether x_k, x_{k+1}, bracket the value a; if they are both below it, take steps to read new card. See par. 4.13.

i:	I 3 (o1), I 6
o1:	$(x_k \leq a < x_{k+1})$ II 1
o2:	$(x_k \leq a, \quad x_{k+1} \leq a)$ I 5
o3:	$(x_k > a)$ Error or I 6 (See par. 8.11)

[1] This output cannot go to I 5 because items read from the card into accumulators 2 and 3 might interfere with the operation of the principal test at the beginning of the next group.

[2] If the output from I 2(o3) is to I 9, the output from I 3(o3) can go to I 7. Under the assumption that w_k/z_k is monotone decreasing, the effect in either case is to read cards to the end of the group without further calculation.

5. Preparation for new card. Clear x_{k-1}, y_{k-1}, z_{k-1}, w_{k-1}; replace x_j, y_j, z_j, w_j by x_{j+1}, y_{j+1}, z_{j+1}, w_{j+1} respectively for j = k-1, k, k+1. The values for j = k+2 are obtained from CT within the 50 add times after the reader is pulsed. Clear Myer (to get rid of previous u_o). Concurrent with I 1 (see under o2)

i: I 4 (o2), I 3 (o2 and o3),

o1: I 1

o2: R_1 if desired. There is no point in this because the entire stage I 5 must be completed within the 50 addition times after the input to I 1 in order to avoid error. Since we must depend on the programming for protection against such an error, the reader interlock is useless.

6. Preparation for new round. Add △a to a Clear the accumulators from which y_a, z_a, w_a, and t_a are printed.

i: VI 7, possibly I 4 (o3)

o1: to I 4

7. Preparation for new group. Add △b to b.

i: I 2 (o2)

o1: I 8

o2: I 2 through dummy (in case I 9 is used.)

8. Initiation of a group. Clear all stored quantities except b (by selective clear). In case b is in the same accumulator with a, a must be cleared without disturbing b. Reset counting stepper if used. Put initial value a_o for a. If $a_o < 0$ we must put a_o for x_k, x_{k+1} to insure that the principal test will not be blocked on the first card. If $a_o = 0$ the stage consists of clearing and resetting only. Concurrent with I 1, I 2.

i: I 7 (o1), I 9 (o1)

o: none or, in case I 9 is not used, I 2.

9. Special card reading. Dummies to separate I 1 from I 8 in case output from I 2 (o3) is needed. In contrary cases can be merged with I 10. Can be omitted in the latter case if also $a_o = 0$, $b_o = 0$. (On the reason for distinction between I 9 and I 5 see footnote to I 2 (o3).)

i: I 2 (o3), I 10

o1: I 8

o2: I 1

10. Start. Sets in initial value of b_o (If $a_o = b_o = 0$ the calculation can start from the Reader Start Button.)

i: Initiating pulse

o1: I 9

II. PREPARATION FOR THE PRIMARY INTERPOLATION.

1. Formation of x coefficient. Form differences of x (in connection with II 2), reading x_{k+2} from CT. If we start with $u_o = 0$ we can begin by putting x_k in position for $f(u_n)$. In some modifications Δa and λ should be set. (See sec. 8.) Concurrent with II 2.

i:	I 4 (o1)
o1:	II 2
o2:	III 1 (in case modification for $u_o = 0$ is used, to III 3)

2. Reception of third difference. The reception of the quantities for the third difference is made a separate stage because it recurs in connection with the secondary interpolations. Concurrent with II 1, IV 1, V 1, VI 1.

i:	II 1 (o1) IV 1 (o1), V 1 (o1), VI 1 (o1).

No output (If the o2 from II 1 is always to III 1, an output from II 2 can be taken to III 2; this would take the place of o1 in III 1, IV 2, V 2 and VI 2, and save four dummies.)

III. ITERATION.

1. Coefficients for x interpolation. Form coefficients A, B, C, D and transmit them at proper times. Concurrent with III 2.

i:	II 1 (o2), III 4 (o3).
o1:	III 2
o2:	III 3

2. Interpolation formula. The multiplications etc. for forming the expression

$$6A + (u+1)\left[6B + u\left[3C + (u-1)D\right]\right]$$

where the coefficients A, B, C, D are from a concurrent stage. Concurrent with III 1, IV 2, V 2, VI 2.

i:	III 1 (o1), IV 2 (o1), V 2 (o1), VI 2 (o1).
	No output.

3. Form $\lambda F(u_n)$. Discrimination on n by counting stepper, and formation of $\lambda F(u_n)$ from $f(u_n)$. (Note that $f(u_n)$ as formed by III 2 is multiplied by 6).

i:	III 1 (o2)
o1:	(if $n \leq N$) III 4
o2:	(if $n > N$) Error, or III 5

4. Inter approximation tests. Form $u_{n+1} = u_n - \lambda F(u_n)$. Make the tests described in pars. 3.25 and 4.13.

i: III 3 (o1)
o1: (if $|\lambda F(u_n)| < \epsilon$) IV 1
o2: (if $\lambda F(u_n) > \epsilon$) Error (III 6)
o3: (if $\lambda F(u_n) < -\epsilon$ and $0 \leq u_{n+1} < 2$) III 1
o4: (if $\lambda F(u_n) < -\epsilon$, $u_{n+1} < 0$) Error
o5: (if $\lambda F(u_n) < -\epsilon$, $u_{n+1} \geq 2$) Error or III 7

5. Reserved for action to increase λ. (Not programmed in basic scheme.)

6. Reserved for action to decrease λ. (Not programmed in basic scheme.)

7. Reserved for action if $u_{n+1} > 2$ (see sec. 8 below.)

IV. SECONDARY INTERPOLATION FOR y.

1. Formation of y differences. Clear $\Delta^3 x$ and auxiliaries in III. Read y_{k+2} from CT and form differences in y. (Third difference received by II 2). Concurrent with II 2.

i: III 4 (o1)
o1: II 2 o2: IV 2

2. Formation of y coefficients. Supply coefficients for forming $6y_a$ by formula analogous to (2.6). Note that it is necessary to take account of the factor 6 in the secondary interpolations, whereas in the primary interpolations it could be absorbed in the value of λ. Concurrent with III 2.

i: IV 1 (o2)
o1: III 2
o2: IV 4
o3: IV 3

3. Secondary transition. Clear third difference. Transfer u_a from Myer for storage while multiplying by 1/6 (in IV 4). Read 1/6 from CT (to be received in Myer in IV 4). Return u_a to Myer after the multiplication. Concurrent with IV 4, V 3, VI 3.

i: IV 2 (o3), V 2 (o3), VI 2 (o3)
No output.

4. Formation of y_a. Receive 1/6 in Myer, and multiply. (Note $6y_a$ is in Mike from IV 2.) Receive and dispose of y_a. Concurrent with IV 3.

i: IV 2 (o2)
o1: V 1

V. SECONDARY INTERPOLATION FOR Z.

1. **Formation of z differences.** Receive y_a from multiplication in IV 4. Read z_{k+2} from CT and form differences of z. Similar to IV 1. Concurrent with II 2.

i:	IV 4
o1:	II 2
o2:	V 2

2. **Formation of z coefficients.** (see IV 2) Concurrent with III 2.

i:	V 1 (o2)
o1:	III 2
o2:	V 3
o3:	IV 3

3. **Formation of z_a.** (See IV 4). Concurrent with IV 3.

i:	V 2 (o2)
o1:	VI 1

VI. SECONDARY INTERPOLATION FOR W AND CLOSURE.

1. **Formation of w differences.** (Analogous to IV 1 and V 1). Concurrent with II 2.

i:	V 3
o1:	II 2
o2:	VI 2

2. **Formation of w coefficients.** Analogue of IV 2 and V 2. Concurrent with III 2.

i:	VI 1 (o2)
o1:	III 2
o2:	VI 3
o3:	IV 3

3. **Formation of w_a.** This is analogous to IV 4 and V 3 with a difference at the end due to the fact that the Eniac is full and that it is not necessary to clear the multiplier. Concurrent with IV 3.

i:	VI 2 (o2)
o1:	VI 4

4. Formation of t_a. This is done according to the formula

$$t_a = t_{k+2} + (u_a - 2)\ \Delta t$$

where t_{k+2} comes from CT and the multiplication by Δt is accomplished by a shifter and repeat switch.[1]

i: VI 3
o1: VI 5

5. Closure. Restore values of w, z, y by reversing difference formation. Reset counting stepper decades, if used. (See par. 3.28)

i: VI 4
o1: VI 6
o2: VI 7

6. Restoration of x-values. This is a separate stage because of connection with Sec. 8 below.

i: VI 5 (o1)
No output

7. Printing. Print (So far as the basic scheme is concerned this can be put on the input line of VI 5. Card number can be put on output cards by the emitter, see sec. 7.)

i: VI 5 (o2)
o1: I 6

7. DETAILED PROGRAM FOR THE BASIC SCHEME

7.1 The schedule of stages discussed in the preceding section could be planned and, except for minor changes, actually was planned, before the assignment of equipment. Likewise the details of the individual stages can be planned more or less in the abstract, with program lines, accumulators, etc., unassigned. To make a complete program it is necessary to put these elements together and to assign equipment in detail. This is more or less of a routine matter; but there are some matters of technique which can profitably be explained here.

7.2 The assignment of equipment involves two main phases: first, the assignment of major items of equipment, such as accumulators, fields of the constant transmitter, steppers, multiplier, etc., also the input program lines, to the various stages; and second, the assignment of individual program lines, digit trunks, and program controls. It will be convenient to consider the input program lines of the stages as belonging

[1] An alternative program in which Δt can be taken from CT and multiplied into u_a by use of the multiplier is laid out in Fig. 24b.

to the first phase; while the program lines connecting the elements of the same stage to each other -- these will be called internal program lines -- belong to the second phase. Then the first phase can be planned on the basis of the schedule of stages in par. 6.4; the second requires, except for the program inputs of the stages, that the programming of all the stages be completed in detail.

7.3 We proceed to the discussion of the first of these phases. Certain incidental observations of importance for the timing etc, of the detailed program will be mentioned as we go along.

7.4 In regard to the accumulators it has already been noted that twelve accumulators, here called storage accumulators, are necessary to store the values of x, y, z, w, from the three preceding cards. These accumulators are therefore not available for auxiliary calculations, except dummies and reversible calculations of the type considered in sec. 5. The latter calculations are performed in the storage accumulators wherever possible in order to save room in the others. The storage accumulators are 1-8, 12, 14-16. The accumulators 9 (Myer), 10 (Mike), 11 (LP), and 13 (Prod) are used in multiplications; they may be used for auxiliary calculations while not multiplying, and one of them, 13, can be used for printing. This leaves accumulators 13, 16-20 for printing. The quantities to be printed are a, b, y_a, z_a, w_a, t_a; of these a and b have only a small number of digits and accordingly they are placed together in accumulator 17; the other quantities are placed in 18, 19, 13, and 20 respectively. (Other quantities, such as card numbers, can be printed through the use of the emitter on the IBM printer or certain decades on the master programmer; these are not used in the basic scheme.) Accumulator 17 then has the same characteristics as a storage accumulator. On the other hand the accumulators 18-20 are not used for storage until the secondary interpolation; and since their digit outlets are not essential to the main calculations, they are eminently suitable as test accumulators for the discriminations in I and III. It is possible to program all these tests on the four digit outlets of accumulators 18 and 19.

7.5 In the above it is assumed that only a ten figure product is kept, so that accumulators 12 and 14 are disconnected from the multiplier. It is in fact supposed that the places and significant figure switches are set at 9.

7.6 As to the constant transmitter (CT) it is not necessary, for the purposes of this report, to plan in detail where the various quantities shall be stored; and it is neither necessary nor desirable -- for reasons of flexibility -- to specify the wiring of the Reader plug-board. All that is essential is to see that the capacity is not exceeded. The following are the quantities we need from each card: x, y, z, w, t, Ø, c. Of these the first four are ten digit numbers, the last three we can suppose are at most five digit numbers, so that 55 digits are used in all. Besides these we need the following constants: 1/6 (as needed for secondary interpolations) c*, a_0, b_0, Δa, Δb, Δt, and λ. Of these Δt and λ are needed only as multipliers, so that they can be taken care of by repeat switches and shifters. The others can be stored in the remaining 45 digits of CT. If a_0, b_0, Δa, and Δb are one digit numbers, c* is five digits, while the 1/6 is stored to ten digits, all these numbers can be set on the switches. It is not, however, necessary to do this; for certain of these numbers can be punched on a master card and locked in the relays by the "reset control" on the reader plugboard. For the sake of definiteness in the present programming, and for no other reason, these numbers are assigned to fields of CT as in Table 7.1.

Table 7.1

A	x
B	y
C	z
D	w
E	t
F	c, φ
G	c*,—
H	Δa, Δb
J	a_0, b_0
K	1/6

It should be noted that the function tables, which are not used in the basic scheme, are available for additional constants in modifications.

7.7 As regards the master programmer, the scheme of par. 6.4 requires five steppers as follows: four for the discriminations of I 2, I 3, I 4, and III 4 respectively, and a counting stepper for III 3. It is not necessary to specify which of the steppers on the Eniac are used for this purpose; they will simply be designated Σ_1, Σ_2, Σ_3, Σ_4, and Σ_5. The rest of the master programmer is free.

7.8 The role of the various pieces of major equipment is shown in the major program schedule, (Figs. 28-31). Here there is a column for each stage, as shown by the heading at the top, and also a column headed 0 for each major process. The zero column is to show the assumed state of affairs at the beginning of the process; the other columns show the changes during the stage. There is a row for each accumulator, including tworows for the doubly-functioning accumulator 17; also rows for other pieces of major equipment, for the input and output program lines, and for the concurrent stages if any. Where a quantity is indicated opposite an accumulator in any column it indicates the contents of the accumulator at the end of the stage or may indicate contents during stage and clearing at end (in the case of the zero-column, at the beginning of the process[1]); clearing of an accumulator is indicated by a dash; nothing at all indicates no change (in that case the accumulator retains what it held in the stage which furnished the input to the given stage, which is not necessarily the next stage to the left.) Opposite CT an entry indicates the quantity or quantities read out during the stage. Opposite MP the use of the steppers Σ_1 to Σ_5 is indicated. In the row for other equipment we have **M** for multiplier, R for reader, P for printer, C for selective clear, I for initiating pulse. An **x** on the left of a column indicates that the equipment on that row is used during the stage other than as dummy; a < (i.e. a half **x**), that it is similarly used in a concurrent stage. Dummies are indicated by a circle o or half-circle on the left; these, of course, are entered on the chart after the second main phase of

[1] In case the process is repeated, the 0 column shows the assumed state of affairs for a typical case. Initially many of the quantities listed may be zero.

the programming. The chart also shows the input and output program lines. Items in parentheses indicate alternatives not a part of the basic scheme. The other entries on the chart are self expanatory.

7.9 In passing to the second phase of the programming we must have plans for the detailed programming of the various stages. It is possible to make such plans, at least preliminary ones, on the hypothesis that each stage is an autonomous piece of computation subject only to the restrictions imposed by the first phase which we have just discussed. In these preliminary plans we designate the input line by i, the output lines by o1, o2, . . ., and the internal lines by (1) (2) (3) (4), . . .; the program controls, digit trunks, etc., could be similarly treated, but we found it feasible to ignore them. Figures 9b and 7b are examples of such preliminary programs. On some of the discrimination programs, certain items have been inclosed in dotted rectangles. These are repeat schedules using the same controls, lines, and digit trunks.

7.10 After the preliminary plans are made we are in a position to assign the program lines and other details. So far as the program lines are concerned this can be done by Table 7.2. This table shows the figures where the final editions of the detailed programs are to be found, the input program lines already assigned in the first phase, the number of internal program lines of two sorts, viz. those which can be made by jumpers and those which have to use a wire in a program tray, and finally the assignment of these lines to program trays (the jumpers are treated as if they belonged to program trays p, q, r, and are indicated by *). For the other items we simply established an order of preference for the stages and assigned the program controls one by one using the wiring diagrams (explained below) and making changes when necessary.

7.11 The final product of all this programming is the set of wiring diagrams in Figures 32-34.

7.12 The wiring diagram shown in Figure 32 gives the planning of the twelve program controls for each of the twenty accumulators. The symbols signify various settings according to their position in the rectangle according to the following code (analogous to Figure 35).

```
a          i
 bcd
           h
```

where the a, b, . . . stand for the following items:

a	input program line
b	function or selective switch setting
c	clear switch setting
d	repeat switch setting
i	stage number
h	output program line

Items in parentheses are possible additions to, but not part of, the basic scheme. Thus by studying this diagram, we know what controls have been used and therefore what controls are still available for further programming. We also know the settings of the switches, the program lines used, and the stage number for any particular control. With this information we may refer to the detailed programs through the stage number and program line of that stage.

Table 7.2

Stage	Figure	Input	No. of Program Lines		Program lines	Jumper
			Jumper	Trays		
I 2	7a	a2		12	e9,j7-j10,k7-k10, h8-h10	
I 3	8	a3	1	5	m6-m10	q8
I 4	9a	a4	1	5	m1-m5	q7
I 5	10	a5		12	1- 10,c9,c10	
I 6	11	a6		1	d8	
I 7	12	a7		1	d9	
I 8	13	a8		3	g7-g9	
I 9	14	a9		0	---	
I 10		a10				
II 1	15	b1	1	3	f1-f3	p1
II 2	15	b2		0	---	
III 1	16	b3	3	3	f4-f6	p2-p4
III 2	17	b4		5	e1-e5	
III 3	18.	b5		1	b10	
III 4	19	b6		8	n1-n8	
III 5		b7			---	
III 6		b8			---	
III 7		b9			---	
IV 1	20	c1	1	3	g1-g3	q4
IV 2	16	c2	3	3	g4-g6	p5-p7
IV 3	21	c3	1	4	c4,e6-e8	q6
IV 4	22	c5		0	---	
V 1	23	c6		4	h1-h4	
V 2	16	c7	3	3	h5-h7	p8-p10
V 3	22	c8		0	---	
VI 1	23	d1	1	3	j1-j3	q5
VI 2	16	d2	3	3	j4-j6	q1-q3
VI 3	22	d3		0	---	
VI 4	24	d4		4	f7-f10	
VI 5	25	d5		5	k1-k5	
VI 6	26	d6		1	k6	
VI 7		d7			---	

7.13 In Figure 33a the wiring diagram is given for the twenty-four controls of the high speed multiplier and the thirty controls of the constant transmitter. Also the connections for the digit trunks are listed.

7.14 For each multiplier control, the coding is as follows:

$$\begin{array}{ll} a_{kl} & i \\ pm & h \end{array}$$

a input program line (M is used to indicate semi-permanent connections between the high speed multiplier and its associated accumulators.)

i	stage number
h	output program line
k	myer switch (with clear indication)
l	mike switch (with clear indication)
m	product disposal switch setting
p	places switch

Here, as in Figure 32, we have a fairly complete listing of the programming and may refer directly to the detailed program through the stage number and program line number.

7.15 For the controls of the constant transmitter, the coding is as follows:

$$\begin{array}{ccc} a & & i \\ & b & \\ & & h \end{array}$$

where a, b, i, h are the same as listed above.

7.16 The digit connections are given in Figure 33a for α, β, γ, δ, ϵ, A, S, for each of the twenty accumulators. The numbers listed show the digit trunk connections which determine the setting of the α, β, γ, δ, ϵ, switches in receiving according to the digit connections on A and S in transmitting. For example if an A is programmed on accumulator 7 to transmit a value to accumulator 2, the β switch must be programmed in accumulator 2 to receive this value as A is on digit trunk 2 and must be received from this digit trunk.

7.17 In Figure 34 the programming of connections 1-11 of program trays a-o inclusive are given. Also trays p, q, r, are added for jumper connections. By checking this diagram it is possible to determine what trays have been used and what trays are still available for programming. For each line the stage number is given together with the program line number as listed in that stage. For example, if in tray g2 the programming is IV 1 (2), g2 is the program line (2) in stage IV 1. If the program line is a stage input, only the stage to which it is an input is listed; the stages to which it is an output may be found from the schedule of par. 6.4 or the major programming schedule Figures 28-31. Thus c3 is input to stage IV 3; reference to the schedule of par. 6.4 shows that it is output of stages IV 2, V 2, VI 2.

8. MODIFICATIONS NEAR A MAXIMUM

8.1 Many cases arise in practice, and these cases are frequent in the calculation of firing tables--where x(t) is increasing toward a maximum, and we wish not only to follow the curve toward the maximum, but to locate, at least approximately, the maximum itself. We consider here modifications of the basic scheme to meet this situation. We suppose that $x < 0$; and that the fit of the interpolation approximation is such that the approximation satisfies the assumptions made for x(t) itself, except for the unavoidable jumps in the derivative.

8.2 The first difficulty to be noticed is that if a is not exceeded by any of the x_k, the principal test will read cards past the maximum, and will continue to read cards until they are exhausted or a new group is ordered. Thus the value of a in question is lost. See Figures 36 and 37. In order to prevent this it is necessary to modify the principal test to stop the reading of cards near the maximum. The arrangement of the principal test is convenient for this purpose. For the output o2 of I4--which orders card reading--comes out of the fourth position of the stepper, while the fifth and sixth positions are free. We can therefore modify the principal test by programming a direct input to the stepper and a further discrimination on the fourth position, with outputs o2 and a new output o4, which stops the reading of cards, coming from the fifth and sixth positions. Of these outputs o2 has the same effect as before, while o4 would be joined to o1, but may if desired, make some special adjustments.

8.3 There are two methods of making the additional test required. If the derivative of x(t) is available, say

$$z(t) = \dot{x}(t)$$

then the card reading can be stopped as soon as $z_{k+1} \leq 0$. Then $z_k > 0$, $z_{k+1} \leq 0$, so that the maximum occurs for $t_k < t \leq t_{k+1}$. This modification of the principal test is shown in Fig. 9b. On the other hand if $\dot{x}(t)$ is not available, the reading can be stopped as soon as $x_{k+2} - x_{k+1} \leq 0$.

In that case we have $x_k < x_{k+1} < x_{k+2} \leq x_{k+1}$; therefore by the law of the mean and our assumption as to the univariance of x, there is a maximum of x(t) for

$$t_k < t < t_{k+2}$$

Examples (see Figs. 36 and 37) show that the maximum of x(t) may be on either side of t_{k+1}. This second modification is shown in Fig. 9c. Note that if we merely stopped the card reading when $z_k \leq 0$ or $x_{k+1} - x_k \leq 0$ the maximum might occur for $t < t_k$. Both of the schemes require additional transceivers.

8.4 By means of the above modifications of the principal test we can cause the Eniac to commence Process II when there is a maximum between t_k and t_{k+2} (or t_{k+1}) even if neither x_k nor x_{k+1} exceeds a. Now if we insist on the monotone case the theory of section 3 applies if there is a root u_a between u_o and the maximum and the assumptions hold for $u_o \leq u \leq u_a$. What happens for $u > u_a$ is irrelevant. Under the assumptions of this section the argument of section 3 shows that persistance of the monotone case can be guaranteed. Hence we shall always get at least theoretical convergence to u_a as long as a root u_a exists. Moreover the maximum is to the right of u_o; for either $u_o = 0$, in which case the matter was settled in the above modification of the principal test, or it is the final u of the previous round, in which case the argument is one of mathematical induction. It follows that there are only two difficulties to be provided for: first the possibility of slow convergence, and second the possibility that a may be larger than the maximum value of f(u).

8.5 We consider first the second possibility. This is an instance of Case 4 in par. 3.13. More specifically we have the behavior illustrated in Fig. 38. The u_n will increase indefinitely with n and eventually will exceed h (defined in par. 3.26). This will cause an output from o5 of III 4; moreover this output will be characteristic of the situation. Thus appropriate actions can be taken by carrying this output to the proper stage.

8.6 If it is not desired to locate the maximum any more accurately than can be done by locating the largest a, in the set of evenly spaced values, for which a t_a exists, and if no special card is to be printed, then the appropriate action in the present case is to terminate the group. This suggests a pulse to I 7. It is, however, necessary to see that the output o3 from I 2 is to I 9 so as to read cards until one comes up with a new φ . Note that the pulse to I 7 automatically clears all the extraneous quantities from the machine. In this case there is no need of a special stage III 7. In all the other cases the output of III 4 (o5) will be to III 7.

8.7 If it is desired in addition to print a special card, more elaborate modifications are necessary. To operate the printer there should be an output (from III 7) to VI 7. The printer output should go through a stepper whose normal output, i.e. from the first position, is the present o1 of VI 7. During III 7 this stepper must be stepped; the output from the second position should then be to I 7 as in the preceding paragraph. Provision must be made for restoring this stepper to normal after an output from the second position; this could be done by setting the counter in the second position at 1, but this runs into the difficulty that there must then be a counter in the first position which cannot be cleared, without modifying the Eniac as above explained, when the stepper is stepped.[1] It may be desirable to clear or adjust the accumulators used for t_a, y_a, z_a, w_a. There must be a special punch to indicate the special card; this may be obtained by using a printing decade in the master programmer, which must be cleared, or by putting a digit in some column which would ordinarily have none (for instance a negative sign in the accumulator for t_a). It is doubtful if these additional complications are worthwhile, and no attempt has been made to program them in this report.

8.8 More interesting is the case where we wish to change the interval Δa in order to locate the maximum more accurately. We suppose that we have available a set of values of Δa of increasing fineness which are controlled by a stepper Σ_6 which is ordinarily in position to deliver the coarsest Δa. The input to III 7 can then begin by subtracting the current Δa from a to restore it to its value in the previous round. We can then step Σ_6 so as to insure that from then on a finer Δa will be used. This general principle requires some changes in detail which will now be considered.

8.9 In the first place we are now proposing to start a new round at the end of process III. It is necessary to restore the differences in x which were generated in II, but not the differences in y, z, w. (This consideration did not arise in the previous cases because these values were cleared by I 8.) For that reason one of the outputs of III 7 must be VI 6. Furthermore dummies must be used so that I 6 is not pulsed until VI 6 is completed.

8.10 In the second place the Δa which was used in I 6 must be stored somewhere in order that it can be subtracted from a. Let us suppose that Σ_6 controls reading out Δa from CT or a function table. Then it must be read into some accumulator and subtracted from a. The reading out of Δa from CT and its reception must be separated in both I 6 and III 7. Alternatively Δa.can be stored in some accumulator or being read out in I 6; in that case it must be cleared in IV 1; moreover none of the accumulators of the multiplier can be used. On this matter see also par. 9.9 ff.

8.11 In the third place when the output of I 6 leads us back to I 4 we may have the situation leading to o3. This is now not an error; it arises because we have decreased a from a previous value. Therefore it is essential that the output o3 of I 4 should be to I 6.

[1] The difficulty can be got around as in footnote to par. 11.4(C)

8.12 In the fourth place the output of the last position of Σ_6 cannot be like the others; otherwise the stepper might cycle back to its first position. This output must terminate the group in the manner explained for the case where the output o5 of III 4 went directly to I 7.

8.13 A program for III 7 embodying these ideas is given in Fig. 39. The two outputs of III 7 are then

o1 to VI 6
o2 to I 6

This program uses the first of the alternatives mentioned in the third preceding paragraph. If the scheme is used the modifications above listed must be made in I 6, I 2, I 4.

8.14 Let us now consider the question of slowness of convergence. Evidently this is a real difficulty here because there is no positive lower bound for α. Consequently it will probably be necessary to use one of the variations in sec. 3 employing a step wise or round-to-round variation in λ. It is therefore appropriate to consider how such modifications can be programmed.

8.15 The first step is to consider the storage of values of λ. In stage III 3 multiplication by λ was achieved in taking $F(u_n)$ from Mike into LP by means of repeat switches and shifters. In the programming of the basic scheme an endeavor was made to keep as many of the controls of Mike and LP open so that there would be room for more than one such program. Inspection of the wiring diagrams shows that there are two transceivers free in Mike and five in LP, and that digit inputs δ and ϵ of LP are free. There are also transceivers and digit inputs free in Myer. This allows for a considerable range of values of λ.

8.16 Leaving aside the question of the capacity in values of λ, let us next consider the control of these values. Evidently these must be controlled by a stepper in which the output of the last position is an error signal or some program, such as an order to terminate the group, which prevents stepping back to its first position. This stepper will be called the λ stepper. If this modification B of par. 3.26 were sufficient it would be possible to combine this stepper with the counting stepper in the manner described under B of par. 3.26. But it is evident that so far as avoiding error is concerned, modification E is more apt to be advantageous. We therefore turn to the control problem for that modification.

8.17 If modification E is used there are two cases where multiplication by λ is employed--viz. in the test in Process II and in Stage III 3. In accordance with our principle regarding resolution into stages, multiplication by λ has now to be a separate stage, we call it the stage II 4. In III 3 we pass to II 4 as soon as $F(u_n)$ is formed in Mike. In II we put $r\beta$ into Mike where r is obtained from, let us say, CT; this r need not be exactly the ratio of successive values of λ but may be an approximation to it. We have reserved the stage II 3 for the formation of $r\beta$; however this may be combined with II 1. The stage II 4 is followed in II by a stage II 5 constituting a test of the inequality $r\lambda\beta>1$; this in turn is followed by a repetition of II 4 with a new value of λ if the test is negative. In order to do this we can give II 4 an output through a stepper to separate this behavior from that desired in III 3. This stepper may be combined with that used for the discrimination in II 5. In fact we can give the stepper a direct clear at the start of II; take the initiation of Stage II 5 from the first position, the negative and positive outputs of II 5 respectively from the second and third positions, and the output of III 3 from the fourth position; at the same time providing a direct input from the first and third positions and a direct clear from the second (thus making these direct imputs separate stages[1]). See also par. 9.14.

8.18 If it is desired to combine modifications B and E the only additional complication is to have a 2 position counting stepper separate from the λ stepper. In that case the first position should be the normal (C 1) position; the second should have its counter switch set at 1 and its output should step the λ stepper as well as continue the calculation[2].

8.19 A modification not mentioned in Sec. 3 is also possible, viz. a test of the inequalities

$$\lambda\left[f(u_{n+1}) - f(u_n)\right] \le u_{n+1} - u_n \le r \quad \lambda\left[f(u_{n+1}) - f(u_n)\right]$$

$$\text{i.e.}\quad \lambda F(u_{n+1}) \le 0 \le r \quad \lambda F(u_{n+1}) - (r-1) \quad \lambda F(u_n)$$

$$= r \lambda\left[F(u_{n+1}) - F(u_n)\right] + \lambda F(u_n)$$

The left hand inequality is redundant. A test of the right hand one could be programmed, but will not be further considered.

8.20 It is evident that these modifications require more equipment. In view of the crowded condition of the wiring diagrams there is doubt as to whether all these complications can be embodied in the basic scheme. But it is not worth while to investigate this further. The reason is that the restriction to three secondary functions with which we started in section 1 is an arbitrary one. In practice we might have a smaller or a larger number; in the former case there will be more equipment available and in the latter case, if we give

[1] See footnote to par. 8.18.

[2] The stepper in par. 8.17, which we shall call for the moment Σ_7, may be provided with counters as follows:

Position	Counter Setting	Output
1	1	II 5
2	1	Direct clear and return to II 4
3	1	III 1 (in place of output from II 1)
4	N	III 4
5	2	Error signal

In this case Σ_7 functions as a counting stepper. If we wish to use the scheme of par. 8.18, it is only necessary for the output of the fifth position to step the λ stepper, put 4 pulses into the direct input of Σ_7, so as to restore it to position 4, and go on to III 4. Of course provision for clearing the stepper must be made as per par. 3.28. Also we must provide that $\bar{\beta}$, $F(u_n)$ etc. are in proper positions.

the data on a set of cards, (see section 10) there may well be. The number of possible modifications is very large. Until some experience is available as to the importance of various modifications it is fruitless to go beyond a discussion of general principles.

8.21 Mention should be made however of what can be done with modification G of par. 3.26. The formation of λ by division would naturally be a part of II. This λ is available as a number, and the formation of $\lambda F(u)$ would be by use of the multiplier. In that event it is easy to incorporate modification (B) also; for we simply multiply λ by r. We can evidently continue doing this indefinitely.

9. A SECOND METHOD OF SUCCESSIVE APPROXIMATIONS

9.1 We consider the following modification suggested by Lt. Col. T. E. Sterne. Instead of computing u_{n+1} by (3.2) let us suppose that the machine has a number of values of Δu, and that

$$u_{n+1} = u_n + \Delta u \qquad (9.1)$$

where Δu is varied stepwise in the following manner. Let the machine calculate (9.1) with the value of Δu used in the previous approximation and then test $F(u_{n+1}) \leq 0$; in the positive case we allow u_{n+1} to stand and proceed to the next approximation; in the negative case we go back to u_n and either take a smaller Δu or, if Δu is already the smallest possible, terminate the iteration. Then, except for the effects of rounding and tolerance in the calculation of F(u), the final u_n will differ from u_a by at most the smallest value of Δu. This process is essentially what a desk calculating machine does in performing division. It bears the same relation to that action of a desk calculator that the process of sec. 3 does to division by iteration. Moreover it is by and large the same process as Horner's method of solving equations, as taught in freshman college algebra, when guessing between approximations is mechanized. The procedure suggested in section 8 for locating a maximum is also essentially a use of (9.1). Authorities on computation are not in complete agreement as to the relative advantages of (3.2) and (9.1) as methods of successive approximation; thus one noted authority on machine computation is said to think that an iterative process of division, using (3.2), would be better than the process actually used in the Divider of the Eniac. Which of these methods is best for the Eniac is probably a matter on which experience alone can decide. In order not to prejudge this matter, both methods are discussed here.

9.2 It is first to be remarked that the adoption of (9.1) instead of (3.2) does not require radical changes in the programming. So far as the basic scheme is concerned the only stages to be changed, except for some matters of detail, are III 3 and III 4. In order to see how the two methods compare, let us imagine n in the discussion of the first paragraph is replaced by n-1, so that we are concerned with a test on $F(u_n)$ rather than $F(u_{n+1})$. Then III 3 reduces to the mere subtraction of a from $f(u_n)$ to form $F(u_n)$[1]. It is convenient to combine this with III 4. In III 4 we make the single binary test $F(u_n) \leq 0$; in the positive case we go out

[1] Here and in the discussion of this section we ignore the fact that the f(u) calculated by III 2 is multiplied by .6. The fact is, however, taken into account on the figures.

to a stage III 5, which adds Δu to u and then goes to III 1; in the negative case we go to a stage III 6 which subtracts Δu from u_n (so as to form u_{n-1}) substitutes for Δu the next smaller value and then goes to III 5. (Note the similarity to the procedure in section 8.)

9.3 Before going into detail on this program let us note what happens in the case of section 8, viz. where we are approaching a maximum. The example of Fig. 40 (which is purposely exaggerated) shows that we may read past a maximum in this case also. One method of avoiding this is for the Eniac to remember $F(u_{n-1})$ and make a further discrimination on $F(u_n) - F(u_{n-1})$. If this is negative it is necessary to subtract $2\Delta u$ from u_n before changing Δu. This will require that the above stage III 6 be split into two stages III 6 and III 8, the changing of Δu being a separate stage. Further there must be a stage III 9 to take care of the case of passing a maximum.

9.4 There are, however, even further complications. When the new $F(u_{n-1})$ is calculated we shall have a test of $F(u_{n-1}) - F(u_{n-2})$; it is therefore necessary to have a value which we can substitute for $F(u_{n-2})$ so that the test in III 4 will work properly. One possibility is for the Eniac to remember $F(u_{n-2})$. On the other hand, since we are certain that $F'(u_{n-2}) > 0$, it is sufficient to put for $F(u_{n-2})$ something, such as x_{k-1}-a, which will have the effect of by-passing the second part of the test III 4.

9.5 Again consider the problem of storing Δu. If we call free acumulators those which are not used for storage throughout the round (this excludes no. 17, as well as 1-8, I2, 14-16) or for multiplying (9, 10, 11, 13) then we have three free accumulators, viz. 18, 19, and 20. Of these one is used for a test, one holds $\Delta^3 x$, and the other must be used to hold $F(u_{n-1})$. Since Δu must be held from one approximation to the next, the only alternative is to hold it in CT or a function table and control it by a stepper. This involves certain difficulties which will be waived for the present. The stepper controlling Δu we call Σ_7. Now since Δu is needed for different purposes in III 5, III 6, and III 9, it is expedient to read it out and store it in, say LP, during III 4. Changing Δu then consists of clearing this Δu, stepping Σ_7 and reading out the new one. The reading out of Δu then has to be a separate stage to which we assign the now vacant number III 3.

9.6 There still remains the problem of controlling the output of III 5. Aside from the possibility of a maximum this has to be to III 1 or IV 1 according as we have or have not exhausted our values of Δu. We cannot make this distinction through Σ_7, as the output there simply reads Δu. To make the distinction let there be a two position stepper - Σ_8 whose output in the first position is to III 1. Then when we signal a readout through the last position of Σ_7, Σ_8, is stepped. The output of III 5 can then be through Σ_8. Now consider the effect of III 9. If this is used we need a further choice between IV 1 and III 7 according to whether we have or have not III 9. This distinction we can make by having an additional stepper.

9.7 With all these features incorporated the revised schedule for III is as follows:

III. Successive Approximations. Stages 1, 2, and 7 same as in the basic scheme.

3. Emission of Δu. Program LP to receive Δu. Go through stepper Σ_7 emit $\Delta_k u$ in kth position, in last position do not emit but send direct input to Σ_8.

i: III 4 (04); III 7 (02)

No output.

4. Interapproximation test. Subtract a from $f(u_n)$ to form $F(u_n)$; test

$$F(u_{n-1}) \leq F(u_n) \leq 0$$

if this is OK replace

$F(u_{n-1})$ by $F(u_n)$ and clear.

Direct clear to Σ_9.

i:	III 1
o1:	$(F(u_{n-1}) \leq F(u_n) \leq 0)$ III 5
o2:	$(F(u_n) > 0)$ III 6
o3:	$(F(u_n) \leq 0, F(u_n) \leq F(u_{n-1})$ III 9
o4:	(all cases) III 3.

5. New approximation. Add Δu to u; go through Σ_8 and Σ_9.

i:	III 4 (o1)
o1:	(Σ_8 normal) III 1
o2:	(Σ_8 stepped, Σ_9 normal) IV 1
o3:	(Σ_8 and Σ_9 stepped) III 7

6. Overdraft adjustment. Subtract Δu from u.

i:	III 4 (o2)
o1:	III 8

8. Adjustment of Δu. Direct input to Σ_7. Clear F(u), leaving $F(u_{n-1})$ unchanged. Clear Δu.

i:	III 6, III 9
o1:	III 5
o2:	III 3

9. Action at maximum. Subtract 2 Δu from u; replace $F(u_{n-1})$ by x_{k-1}-a. Direct input to Σ_9

i:	III 4 (o3)
o1:	III 8

9.8 The following minor changes are also necessary. Since we start with $u_o = 0$ we need not save u_a in VI, but should clear it. This effects some simplification there. In II 1 we must put x_{k-1}-a in place of $F(u_{n-1})$, and, if we elect to go from II 1 direct to III 4, also x_k in place of $f(u_n)$. Provision has to be made for restoring Σ_7, Σ_8, and various other instruments concerned with controlling Δu (see below) before a new round is begun.

9.9 It is evident that the above program simplifies greatly if we do not have to worry about approaching a maximum. In fact we leave out everything involving $F(u_{n-1})$ or Σ_8 and we combine III 6 and III 8. Under the same circumstances, however, the program of sec. 3 also simplifies greatly; in fact the discriminations (b) and (c) are safeguards which become unnecessary if we are sure of definite limitations on α and β . To obtain comparable protection in the present case we should need to retain both tests in III 4 even though o3 were only an error signal. A detailed program of III 4 in such a case is given in Fig. 41. The complexity is of the same order of magnitude as in the basic scheme. Inspection of the above schedule shows further that we need more stages, but that some of the stages are very simple. When the entire schedule is programmed in detail, it is probable that it will be of the same order of complexity as that in sec. 3 and 8. At least the a-priori difference is not great enough to be decisive.

9.10 We consider now the difficulty of storing and controlling the various values of Δu. The situation is the same with respect to the values of Δa in par. 8.10 except that we shall presumably need a finer mesh for the smallest value of Δu.

9.11 One method of storing these Δu's is to put them as separate entities in CT. Since they are apt to be one digit numbers, this is possible so far as number of digits is concerned. On the other hand a rather unpractical number of shifters and digit inputs is likely to be necessary.

9.12 Another scheme is to store the values in a function table. In the basic scheme we have deliberately avoided using the function table, on the ground that the scheme would be more useful if the function tables were free to be set for another problem while the inverse interpolation was going on. If the function tables are used, note that we do not have a free accumulator available to store an argument. However five values are available from a function table without an attached argument accumulator; in that case the function table acts as if it had the argument zero. The five possible outputs correspond to the five available positions of Σ_7.

9.13 Since the values of u are most likely to be units in different decimal places, it should be possible to generate them by subtract correction pulses and shifters alone. A scheme for doing this is shown in Fig. 42. This gives, in fact, a slight generalization, in that with the stepper in the kth position, k=5, the value of Δu generated is, with the upper row of stepper outputs,

$$\Delta_k u = \sigma_1 \, p(\sigma_2 \, q)^{k-1}$$

where p and q are integers from 1 to 9, and σ_1 and σ_2 are powers of 10 which are introduced respectively by the x input in Aux. 3 and the combination of the β input on Aux. 3 with the α input in Aux. 4. Here Aux. 1, Aux. 2, Aux. 3, and Aux. 4 are any four "auxiliary" accumulators; we could, for instance, identify Aux. 3 with Mike and Aux. 4 with LP. The program uses eight dummies, one transceiver and two receivers in Aux. 3 and two transceivers and two receivers in Aux. 4. The program could be modified in various ways. The dummies do not have to be in specific accumulators Aux. 1 and Aux. 2 but may be placed anywhere in the machine where dummies are available. Again if the shifter in the α input to Aux. 4 is set so as to multiply by σ_2 / σ_1 the digit inputs α and β of Aux. 3 can be the same; if further an additonal dummy is used to isolate the stepper input lines (1) and (3) can be combined. With reference to the large number of dummies note that four dummies can be made available by the expedient mentioned in the discussion of stage II 2 in sec. 6.

9.14 If a free accumulator is available there are several other possibilities. We can then have a function table with argument; the argument being increased by a unit from III 8. Again we can keep Δu in an accumulator and change it as needed by having III 8 pulse the program in the dotted rectangle of Fig. 42. In either case the stepper is used to discontinue the process when the smallest value of Δu is reached.

9.15 The program in Fig. 42 is useful in another connection. For its essential characteristics are that it generates a sequence of quantities in geometric progression. With appropriate modifications it is a suitable program for the stage II 4 of par. 8.17. For this purpose we shall omit the subtract correction pulses on the input line and put the transfer on line (1) on the input line. The transfer can be repeated p times; in that case the input to the stepper should come on a separate line which can be the output of the transfer. This can be the new line (1). Since the λ's increase as the stepper is stepped, and since, for reasons of accuracy, it is preferable to have $\sigma_2 q < 1$, the outputs from the first five positions of the stepper should be reversed as shown on the second line under the stepper. Then $\sigma_1 p$ is the largest value of λ, $\sigma_2 q$ is an approximation to 1/r. If this scheme should be used it would be expedient to make the test in II 5 a comparison of

$$\lambda\beta < \sigma_2 q.$$

If the program in the dotted rectangle is made a separate stage, say II 6, then the right side of the inequality can be formed by putting 1 through II 6. However this matter will not be gone into further.

9.16 From the above it is evident that the calculation based on (9.1) is comparable in complexity with that based on (3.2) with a stepwise variation in λ. If the latter calculation can be made satisfactorily with a single value of λ it is probably the simpler of the two.

9.17 In regard to the question of accuracy, note that the present method is accurate to within the smallest value of Δu, unless F(u) is so very flat that its values are indistinguishable over an interval greater than this limit. The accuracy of the former method is controlled by the inequalities (3.17) and (3.19). If the accuracy demanded in $u_a - u_n$ is not too great, there is likely to be room for a suitable choice of ϵ in the method of sec. 3.

9.18 In regard to the timing requirements, observe that for $\Delta_k u = 10^{-k}$, $k = 1, 2, 3, 4, 5$, the maximum number of approximations for a round on the present method will be 5, the maximum 50 (not counting the calculation at zero). This is for a final tolerance of 10^{-5}; for greater accuracy the number of approximations would be greater. The number of approximations for the former method was discussed in sec. 3.

10. COMPOSITE INTERPOLATION

10.1 Let us now consider what can be done in case the number of secondary functions exceeds the number permissible in the basic scheme. Such cases may arise in connection with firing tables where a large number of differential effects have to be considered.

10.2 Since there is not room in the Eniac to store all the quantities needed, one suggestion would be to have more of these quantities available on the cards. Differences etc. could be placed on the cards by IBM machines while the Eniac is being set up. This would require that the data for any single calculation be distributed on several cards; such a collection of cards will be known as a **set.** The different cards of a set can be distinguished by a stepper functioning like a digit selector on an IBM machine. It is natural to suppose that all the data for a single function are on the same card. In that case the card bearing the information on x will be called the **primary card,** the others **secondary cards.**

10.3 The simplest assumption is to suppose that all the differences of each function are given on the appropriate card, so that is unnecessary to calculate differences or to retain the information from more than one function. In that case there may be two functions on a card. Even so the system is hardly practical. For, under the hypothesis that the content of a set of cards is too great for the Eniac to remember, it is not possible to have more than one value of a between two successive x_k's. Not only is this an undesirable restriction, practically cutting out, as it does, any possibility of the adjustment considered in sec. 8, but even in cases where it appears to do no harm, it may be difficult to be sure of that fact over any considerable range. If the method is used we should therefore have an error signal in I 4(o3). Where the hypothesis is not fulfilled the method competes with the basic scheme. If the number of sets in a run is n, while the number of values of a is m, and if we suppose the calculations take approximately one card cycle, then the time for a run in the basic scheme is m+n card cycles. (This neglects the effect of delays due to discriminations between card readings in 1. Of these I2 and I 4 affect both methods alike.) To do the same job with the new method requires 2 cards per set, hence 2n card cycles for reading; in addition, unless we make the above restriction on values of a, the secondary card must be held till completion of the interpolation andprincipal test before ordering a new primary card, and there will be additional delay if a second round is ordered without reading; hence the total time will be somewhere between 2n and M+2n card cycles[1]. The principal advantage of the method will be the reduction in storage accumulators; indeed if we impose the above restriction on the a's we can get along without any. Here however the considerations adduced, together with the extra card preparation are regarded as decisive objections, and no attempt is made to program the

[1] In the basic scheme computing and card reading alternate: in the alternative scheme the primary and first secondary interpolations can be simultaneous with the reading of the second card in the set. Since the processes V and VI only take roughly 150 add times the lower limit of 2n card cycles might be attained if m ≤ n. In any event the number of card cycles is at least the larger of the two numbers m+n and 2n. The upper limit, m+2n, is likewise on the hypothesis that there is no delay due to the last two secondary interpolations. Any inaccuracy in these assumptions will make the above comparisons even more unfavorable for the new method.

method in detail[1]. That may very well be done later, and then the result of trail will be the answer as to practicality.

10.4 Another possibility is to make a run with the basic scheme first, and then to use the output cards of this run as primary cards for a new run. These can be interspersed among the sets of secondary cards, according to the value of t, by the collator. This form will be called **composite interpolation.** For the primary cards give us $t = t(x)$, while the secondary cards give $y = y(t)$ for each secondary function y; the output is $y = y(t(x))$, i.e. the composite product of the functions y(t) and t(x). This is a problem which may well be of some interest in itself, when the cards giving t(x) are a given set of cards, which may or may not be the result of a previous inverse interpolation. We shall therefore pause to examine this composite interpolation.

10.5 It should first be noticed that composite interpolation is not an efficient process for the Eniac. Indeed, since all the Eniac has to do is to perform a small number of direct interpolations per card cycle, it will be idle a large proprotion of the time; if there are 5 interpolations, each taking 60 add times, it will be idle 90% of the time. On the other hand such interpolation is an efficient process for the IBM relay multipliers. In that case it is not necessary to collate the primary and secondary cards; the machine has two feeds and can do its own collating as it proceeds. One of the feeds of course delivers into the output; if we use this for the primary cards there must be room on those cards for receiving the answers. The machine will make a third order interpolation in one double card cycle. If we use **m** and n as before, and put **k** functions on one secondary card, the time for a run is $2km + n$ card cycles. There are difficulties in taking $k > 1$. Note that a single card cycle on the relay multipliers is the same as a card cycle on the Eniac[2].

[1] This could be done by modifying the basic scheme. The principal changes would be to provide for the discrimination as to card kinds and to read the differences from the cards on a fixed set of accumulators. If we suppose, for simplicity, there is only one function per card (this assumption is in agreement with the idea that the principal advantage of the method is to save storage accumulators) we can use the fourth alternative. The following outline of changes may be of interest for the light it sheds on general principles. The stages I 3, I 5 would be replaced by other stages: the stages II 1, IV 1, VI 1, VI 5, VI 6, would be omitted. The emission of differences and their multiplication by numerical coefficients would be combined with III 2, thus reducing III 1, IV 2, V 2, VI 2, to dummies: these dummies could be avoided by giving III 2 an output through a stepper or by splitting it into parts and using additional controls on the multiplier. The processes IV, V and first half of VI are then all just alike except for disposal of the output: accordingly, if this output is left in a fixed position, they can be combined into a single process VII. To discriminate the card kinds we need three steppers Σ_a, Σ_b, and Σ_c: of these Σ_a is a two position stepper acting as flip flop, which is in first position for reading and in second position for interpolating; while Σ_b and Σ_c are stepped together to indicate the kind of card. The outputs of Σ_a are to Σ_b and Σ_c respectively. The first output of Σ_b orders the principal test I 4; the others go to I 9. Σ_b could be a two position stepper. The principal test must compute x_k and x_{k+1} from the differences; the output o1 steps Σ_a and goes to III 1 or III 2; the output o2 goes to I 9. As to Σ_c, output in the kth position ($k > 1$) should first dispose of the result of the (k-1)st calculation, then go to VII; in the first position it should close out the whole preceding round, reset Σ_a, and go to I 4. The output of III and VII should go to I 1. The reader interlock should be used as in the scheme for composite interpolation below. It is evident these changes are extensive enough to justify a reprogramming.

[2] The most recent information on the Eniac card reader requires a modification in this and other statements regarding the Eniac card timing. All the rough estimates in this report are based on a card cycle of 0.6 second, i.e. 100 cycles per minute, which is the length of a single card cycle on the IBM Relay Calculators. The most recent Eniac report states the reader will take 120 to 160 cards per minute depending on circumstances.

10.6 Although the process is relatively inefficient for the Eniac, yet for all that the Eniac can do the job faster, although not by a large factor. Furthermore there may be special reasons for using the Eniac. There is thus some interest in seeing, in a general way, how composite interpolation can be programmed; but there is hardly any point in going into detail. The principal case of interest is where the Eniac is already set up to do inverse interpolation by the basic scheme (since it will hardly pay to set up the Eniac specially for this problem), and it is desired to make modifications as simply as possible so as to perform composite interpolation.

10.7 Since the Eniac does not have but one card feed it is necessary to collate the cards. There are two methods of doing this: either the primary card is inserted so that all the data necessary for the calculation are stored in the Eniac from preceding secondary cards, or else the primary card is inserted so that additional data from succeeding cards is needed. In the former case we are restricted to three secondary functions,[1] and a primary card simply gives the signal to compute using the data in the storage accumulators; there is no limitation on the number of consecutive primary cards. In the latter case we may increase the number of secondary functions to four, but the programming is more difficult and there is trouble if there are consecutive primary cards. We assume the former case. We also suppose that the secondary cards contain the value of each of three functions, which we call y(t), z(t), w(t), for a given t which plays the role of t_{k+2}; further that they contain the appropriate $t_k = t_{k+2} - 2\Delta t$ for use in collating. The primary cards will be supposed to have both u_a and t_a.[2]

10.8 The following are the required changes in the basic scheme. The stages I 3, I 4, I 6, II 1, all of III except III 2, VI 4, VI 6, are omitted; of these VI 4 is simply short circuited, (VI 4 may be retained to transfer t_a), while I 3 and I 4 are replaced by other stages (see below). The accumulators 1, 2, 3, must now be used to store y_{k+2}, z_{k+2}, w_{k+2}--as there are four, rather than three, values of each function to be stored. We change I 5 so as to cycle the four values, rather than the three, and in IV 1, V 1, VI 1 we read these values from the storage accumulator rather than CT. The item involving **a** in I 7 can be omitted but is harmless. The reader interlock can be used; indeed we can pulse R_i at the beginning of I 5 or IV 1, and go to R_1 from the end of I 5 and VI 7. For the new stages I 3, I 4 we have:

I 3 Discrimination between primary and secondary cards.

(This can be done by reading from some card column into the direct input of a stepper.)[3]

i:	I 2 (o1)
o1:	(primary card) I 4
o2:	(secondary card) I 5

[1] We have only twelve storage accumulators. In the first of the two methods all four values of each function must be stored, in the second the fourth value might be stored in CT, as in the basic scheme.

[2] Note that the basic scheme does not provide for printing of u_a. But additional quantities can be printed by transferring the contents of accumulators 14, 15, or 16 to Mike or LP before printing, so as to free these accumulators for additional printings and restoring them in I 6.

[3] In this way we could also distinguish two or more different sorts of secondary cards.

I 4 Initiation of Computation. Replace a in acc. 17 by x from the card; read u from card into Myer, store t_a for printing.

i:	I 3 (o1)
o1:	IV 1
o2:	I 1 (i.e. R_i)

A frill which can be used if desired is to keep track of a by use of I 6 as in the basic scheme (going to I 6 from I 4) and make in I 4 an equality discrimination on x $\gtrless$ a. A program for the revised I 5 is shown in Fig. 43. In this figure we have departed from our usual conventions in that we have numbered the program lines not consecutively, but so as to correspond with these in the original I 5; the program is the same as the original I 5 except for the accumulators 1, 2, 3 and the items marked #.

10.9 Thus the basic scheme can be converted into a scheme for composite interpolation by making a relatively small number of changes. The time for a run is m+n card cycles, as opposed to 6m + 3n cycles (assuming k = 1) in the relay multipliers. In case the primary cards come from a previous interpolation the time is of the same order of magnitude as that for repeating the basic scheme. Under certain specialized conditions the modification may well be useful. Note that there is no delay between card readings.[1]

11. CONCLUDING REMARKS

11.1 The major part of the task outlined in the Introduction, viz. the programming of the basic scheme, was completed in sec. 7. The program is complete except for details mentioned in par. 11.5 below.

11.2 As stated in pars. I.3 and 3.26, this basic scheme was not designed specifically for a particular problem, but as a basis from which modifications could be made for various such problems. However with the rather slight modification in par. 3.26 (a) it is suitable for the first step in the conversion of a set of trajectory cards into a firing table for an antiaircraft gun. This will be gone into further in another report.

11.3 Of the many modifications which may be applied to the basic scheme a few have been discussed extensively in secs. 3, 8, 9, 10. These modifications were also motivated by possible utility in connection with firing tables. However it is not feasible to program them in the same detail--with wiring diagrams etc. --as the basic scheme. That is left to be done later when the extent of the demand for such changes is clearer.

11.4 Besides these modifications there are various others which have been mentioned only incidentally or not at all. For convenience of reference a number of these changes are listed below, together with cross references and such comments as it seems appropriate to add to those already made.

(a) If there are less than three secondary functions process V and possibly also portions of IV and VI can be omitted. This leaves room for other modifications. If the number is greater than three see, sec. 10.

[1] Cf., however, the footnote to par. 10.3.

(b) A change in the interpolation formula was mentioned as a possibility in par. 1.4. This has not been pursued farther in this report; however a separate report dealing with one such modification is under consideration.

(c) It was supposed (par. 1.9) that there are two cards before the first value of a and two after the last value of a. This is so that, when the principal test orders a calculation, all four values on which it is based may be bona fide. If the cards do not satisfy this condition, extrapolations will have to be made. Nothing has been done to provide for having the Eniac take special action in the eventuality that the cards do not satisfy the condition. In the basic scheme, if this eventuality occurs at the beginning of a group, the Eniac may print an erroneous result; whereas if it occurs at the end, the value of a will be skipped. The occurence of an error at the beginning may be prevented by taking the output o1 of I2 through a stepper whose first position has an output to I5 and a counter switch set at 3, while the second position has an output to I3 and a counter set sufficiently high to allow the passage of all the cards in the group. This stepper must be reset by I8. (On the difficulties of resetting a stepper with counter see par. 3.28)[1] If this is done we do not need the initial a_o in I8; also, if $b_o = 0$, we can start with the reader start button.

(d) One or more function tables can be used to store additional constants. For some incidental remarks in regard to this see par. 9.12.

(e) The modifications C, D, F, of par. 3.26 are not further considered. The case $F'' > 0$ can be reduced to $F'' < 0$ by the artifice considered in par. 1.5. (See also (o) below.)

(f) As remarked in par. 4.16, the auxiliary test I 3 is included as an example of additional tests which might be included in I. Such tests are likely to be different for each particular application.

(g) There are two alternative forms of the group test, viz. Figs. 7a and 7b. The second differs from the first in that it saves an addition time at the expense of an extra dummy. (See footnote to par. 4.18)

(h) Various alternatives are given for the output of I 2, I 4, I 5, I 7, I 8. A choice between these alternatives is required if the modification of sec. 8 is used. (See pars. 8.2, 8.6, 8.11)

(i) The a_o put in I 8 is to prevent malfunctioning of the principal test at the beginning. A more adequate protection is discussed in (c) above. Likewise b_o in I 10 is necessary to prevent malfunctioning of the group test. Note the alternatives in regard to starting--mentioned in par. 6.4 under I 9, I 10.

(j) In the basic scheme we go to some trouble to preserve the final u of one round for use in case we have a further round without card reading, as the u_o of the next round. This is to keep the u's increasing throughout the group. (cf. par. 3.20). However that may be an unnecessary refinement, and it is useless if we read a new card for every round. If we start with $u_o = 0$ we can make slight changes in II 1 and VI 4.

(k) As noted under II 2 in par. 6.4 we can give II 2 an output and save four dummies. This saving of dummies may be important if the modifications in sec. 8 and sec. 9 are used. (See the discussion of Fig. 42 in par. 9.13.)

[1] In this case the difficulty in regard to the stepper counter can be circumvented as follows. Let the connection from the regular stepper input to the counter be severed (e.g. by disconnecting tube A-43 on drawing PX 8-304), and let a line be taken from the first stepper output, or some properly insulated output connected with it, to the digit input. (cf. (u) below.). Then the counter controls the stepping to 2nd position, when this happens the counter is cleared and receives no more inputs. To restore the stepper it is only necessary to give it a direct clear.

(l) In par. 6.4 stages III 5 and III 6 were reserved for increase and decrease of λ. Of these III 6 would be needed in connection with modifications (C), (D), (F) of par. 3.26. The stage III 6, therefore, has not been further considered in this report. (cf. (e) above). As for III 5, the appropriate action would be simply to step the λ stepper. If the λ stepper is at the same time a counting stepper, this is done automatically in III 3, and III 5 is unnecessary. If the λ stepper is not also the counting stepper, the stage III 5 consists simply of a direct input to the λ stepper and continuation of the calculation. In that case we have the situation of par. 8.17, where the multiplication by λ constitutes a separate stage II 4. This stage consists of separate actions, controlled by the λ steppers. A method of doing this is discussed further in par. 9.15. If the counting stepper is used it should be connected as in par. 8.18 (cf. footnote to par. 8.17). Note that the output of the last position of the λ stepper must always be an error signal (to prevent cycling back to the first position). Further, when a counting stepper is used, the above arrangement allows the calculation of one additional u with the old λ before making the change.

(m) The basic scheme uses the same interpolation formula for the secondary interpolation as for the primary. With some additional complications a different one could be used. This is not further considered here.

(n) Various changes could be made if the number of significant figures were different. In the basic scheme it has been assumed that the places switch on the multiplier are set at 9 (par. 7.5) and that the accumulators and CT groups holding **x, y, z, w** are full.

(o) As explained in par. 1.5 the basic scheme was planned primarily for the case that $\dot{x} > o$, $\ddot{x} < o$. If these derivatives have fixed signs consistent with this assumption, it is only necessary to change the signs of Δa, Δt, or both. (See the footnote to par. 1.5.) This entails some changes in the text of I 8, 9, 10 and, if $\Delta a < 0$, reversal of the inequalities in the principal test. Likewise we can have $\Delta b < 0$ if we make the proper changes in I 10 and I 2.

(p) The stage VI 4 was planned for the case that Δt is a number of only one significant digit. However by an additional use of the multiplier--whose controls are relatively little used--we could handle an arbitrary Δt; if this Δt has not more than 5 digits it can be stored in field G_r of CT. (cf. Table 7.1 in par. 7.6). A program for VI 4 under these conditions is given in Fig. 24b; it is not essentially more complicated than Fig. 24a.

(q) As was pointed out in par. 7.6, considerable latitude is possible in regard to the arrangements of CT. In the scheme of Table 7.1, a_o and b_o were set on the switches so as to be available for the initial entries in I 10 and I 8; the other constants can then be put in by a master card. But if $a_o = b_o = 0$, the switches J could be used for other constants.

(r) Arrangement can be made to print u_a as well as t_a on the output cards. See footnote at end of par. 10.7. Card numbers can be printed on the output cards by means of the emitter on the printer plugboard.

(s) The basic scheme contains no provision for checking whether the input cards are consecutive. If one or more input cards were omitted erroneous results would be obtained. Such a check seems to be desirable for some firing tables work. We can obtain it by modifying I 5 so as to read t_{k+2} into an accumulator, preferably one of those in the multiplier; then subtracting this from the t_{k+2} of the next card and checking for comparison with Δt as part of the auxiliary test. If the difference is found too large, action can be taken to throw an error signal or terminate the group.[1]

This can be done using the same test apparatus as in I 2. If we wanted action for too small a difference--it would be a gross error--we should require other equipment; but if the auxiliary test were omitted the test apparatus of I 3 would be free for such a purpose. Of course this procedure requires that Δt be available, say in CT; it is also necessary to make special provision for the initial t. The latter can be entered, perhaps by calculation from b, as part of I 8; or we can make arrangements by a stepper to by pass the first card. This test, as a safety measure, is of the same nature as (c), and, if a stepper is used it depends, like that test, on the possibility of clearing the stepper and its counter at end of every group. If it is desired to by pass the first three cards the stepper can be combined with that in (c). Naturally care must be taken to see that the t's so entered are properly cleared in every eventuality.

(t) In the basic scheme it was assumed that a_o is constant. In firing table work cases occur in which it is convenient to have a different a_o for every group. This can be done,. of course by interpolating a calculation of a_o from b in I 8. An alternative procedure is to have a master card in the front of every group; then we can have a whole new set of the constants a_o, Δt, Δa, c for each group. Further we can get a different initial λ for each group by using a column in the master card to control the λ stepper through the direct input.

(u) Another method of controlling values of λ without discriminations is of some interest. If the connection from the regular stepper input to the stepper counter is severed (say by removing the tube A-43 on the drawing PX 8-304), the stepping of the stepper can then be controlled by the counters, but the latter count the inputs to the direct decade input without regard to the number of regular inputs. Thus if we take a program pulse to this decade input from I 7, and if the switch settings are n_1, n_2, . . ., then we have the first value of λ for the first n_1 groups, the second for the next n_2 groups, and so on, regardless of how many times a λ is called for in each group. If we provide for clearing the stepper and counters at the end of a group we can get a similar control by rounds by taking a program pulse from I 6. Furthermore by use of a second stepper and direct inputs we can combine these controls in various ways. This method of control may be more convenient than those in (t), par 3.26 and sec. 8. in cases where a single value of λ is known to suffice for a whole series of groups or rounds.

11.5 Although the basic scheme is reasonably complete, there are nevertheless certain details which have been left undetermined. There are also certain items in the internal settings of the machine which must be taken care of before the scheme is actually set up on the Eniac. A list of such details is as follows:

(a) Wiring of the IBM plug boards. This must include provision for a master card to contain c*, Δa, Δb(cf. par. 11.4 (q)).

[1] The group can be terminated by a pulse to I 7 if the output from I 2(o3) is to I 9.

(b) The assignment of steppers to units of the master programmer. These steppers are here simply designated $\Sigma_1, \ldots, \Sigma_5$. Of these Σ_1, Σ_2, Σ_3, Σ_5 are without counters. Since the MP has only two steppers (A and F) without counters, it will be necessary for the maintenance crew to disconnect two others.

(c) The settings of the significant figure switches.

(d) The specification of the various adapters and the locations of decimal points. Places where shifters are obviously likely to be needed are indicated on the wiring diagrams, but others may be needed if there is any irregularity about the decimal points. Particular care should be exercised in operations involving accumulator I 7, which plays a double role. Also it is necessary for the successful operation of III 4 that the subtract correction pulse in LP go into a decade off by a deleter in the ϵ terminal of accumulator 19.

(e) The settings of k and shifters Fig. 18 and Fig. 24 causing multiplication by λ and Δt respectively.

(f) The settings of the printer switches and the provision, if any, for printing card numbers or other constant data on the output cards. For the basic scheme only the accumulators 13, 17-20 are used for printing; of these 13, 18-20 are to print 10-digit numbers, while 17 is to print two separate 5-digit numbers.

(g) Considerations relative to the loading of the Eniac digit trunks and program lines.

(h) Provision for clearing stepper counters if and as needed (cf. par. 3.28.)

(i) Verification that the switches are correctly set, and that the data cards satisfy the hypotheses of sec. 1. (See especially par. 11.4 (C).)

11.6 The process of inverse interpolation here programmed for the Eniac may also be carried out on the IBM relay multipliers and the BTL relay machines. Those machines, however, will be much slower (cf. par. 10.6). Consequently if the job is tried on those machines more attention must be paid to speed of convergence. At best, however, they will still be much slower than the Eniac. It is considered advisable that the Eniac should be used for firing table calculations where the computations come in large masses to be done at one time; whereas the other machines can and should be used for computations coming up repeatedly in smaller amounts.

Haskell B. Curry
Haskell B. Curry

Willa A. Wyatt
Willa A. Wyatt

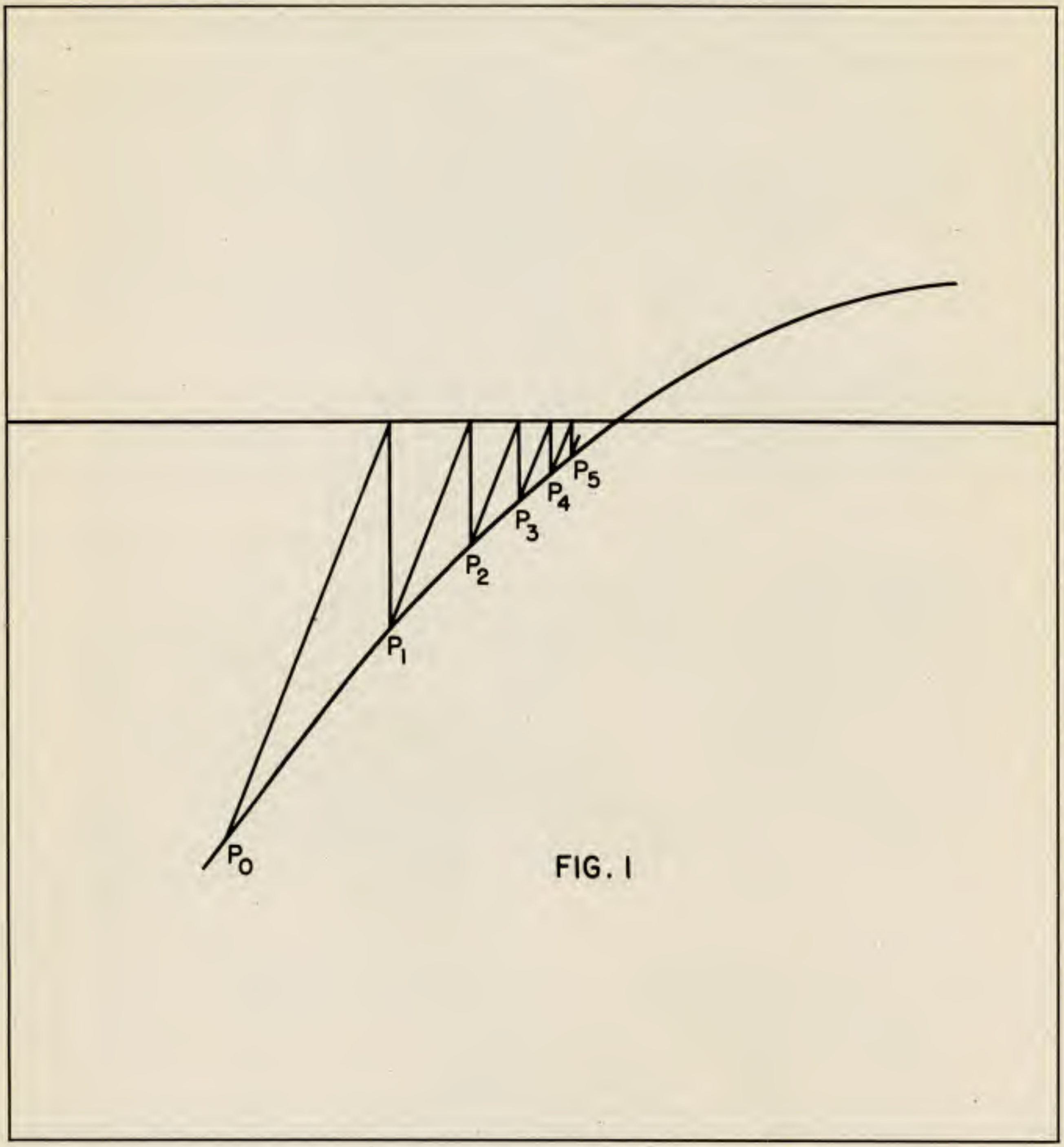


FIG. 1

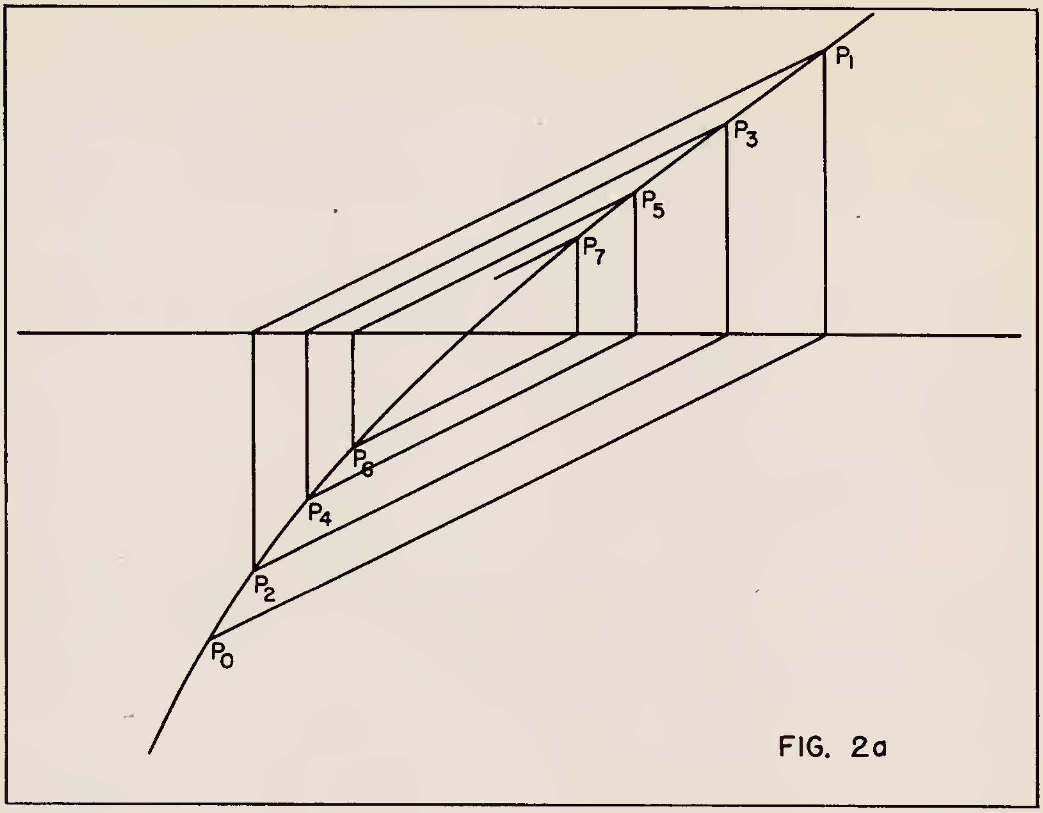


FIG. 2a

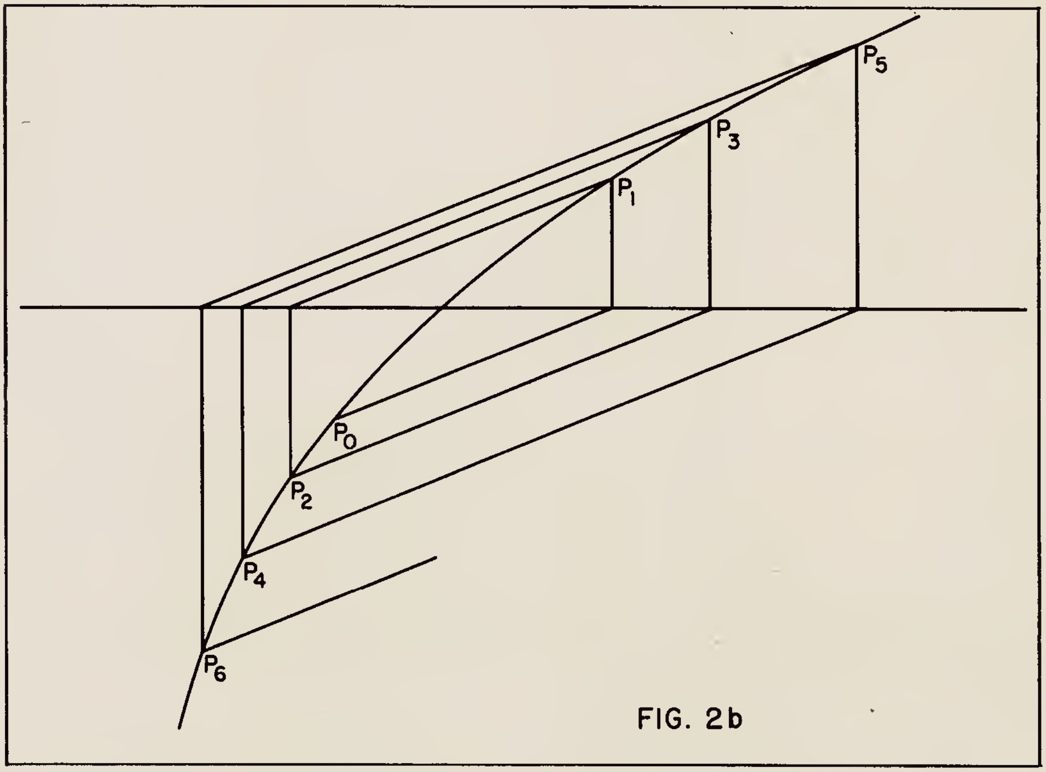


FIG. 2b

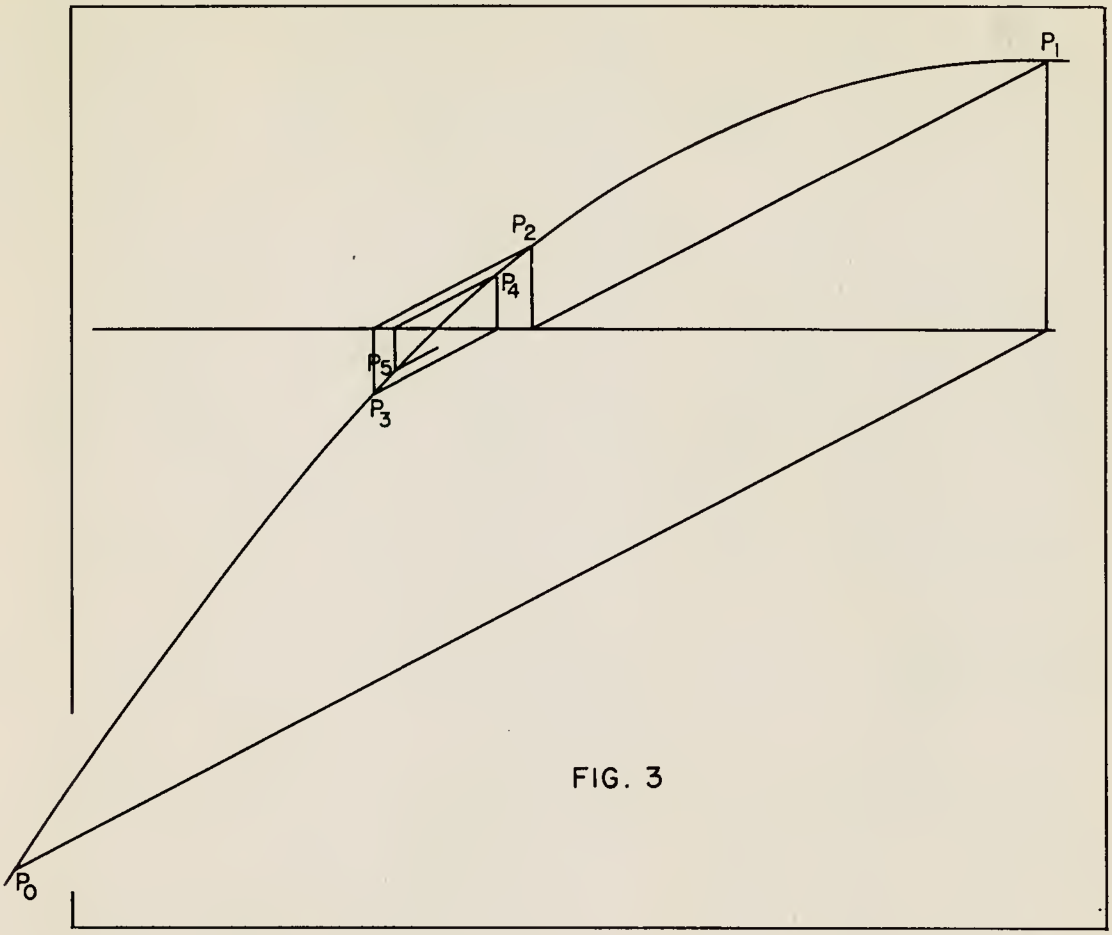


FIG. 3

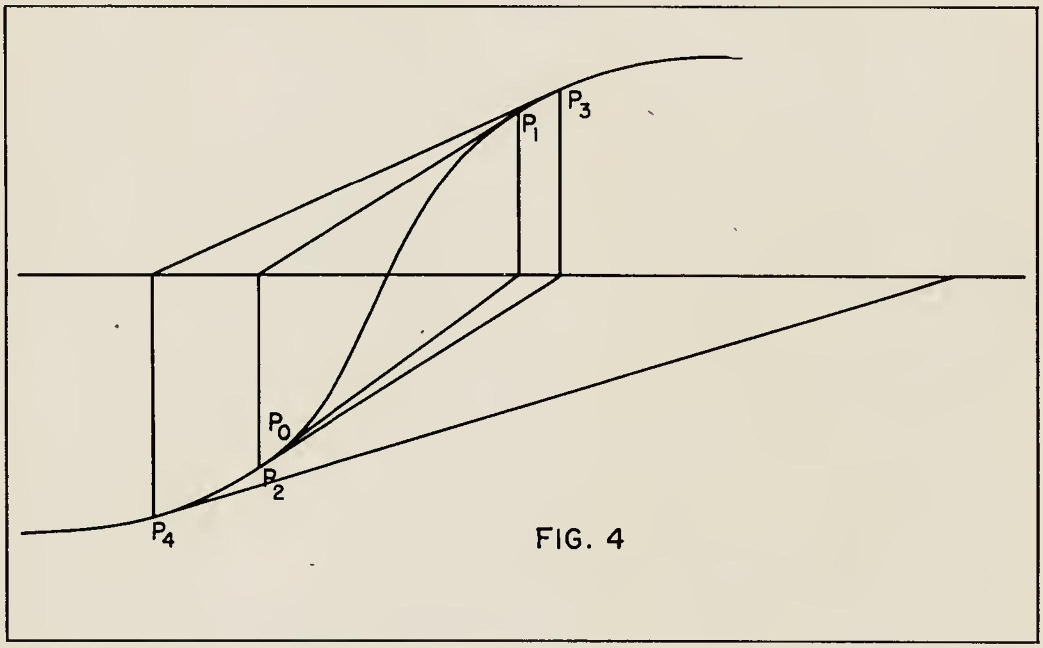


FIG. 4

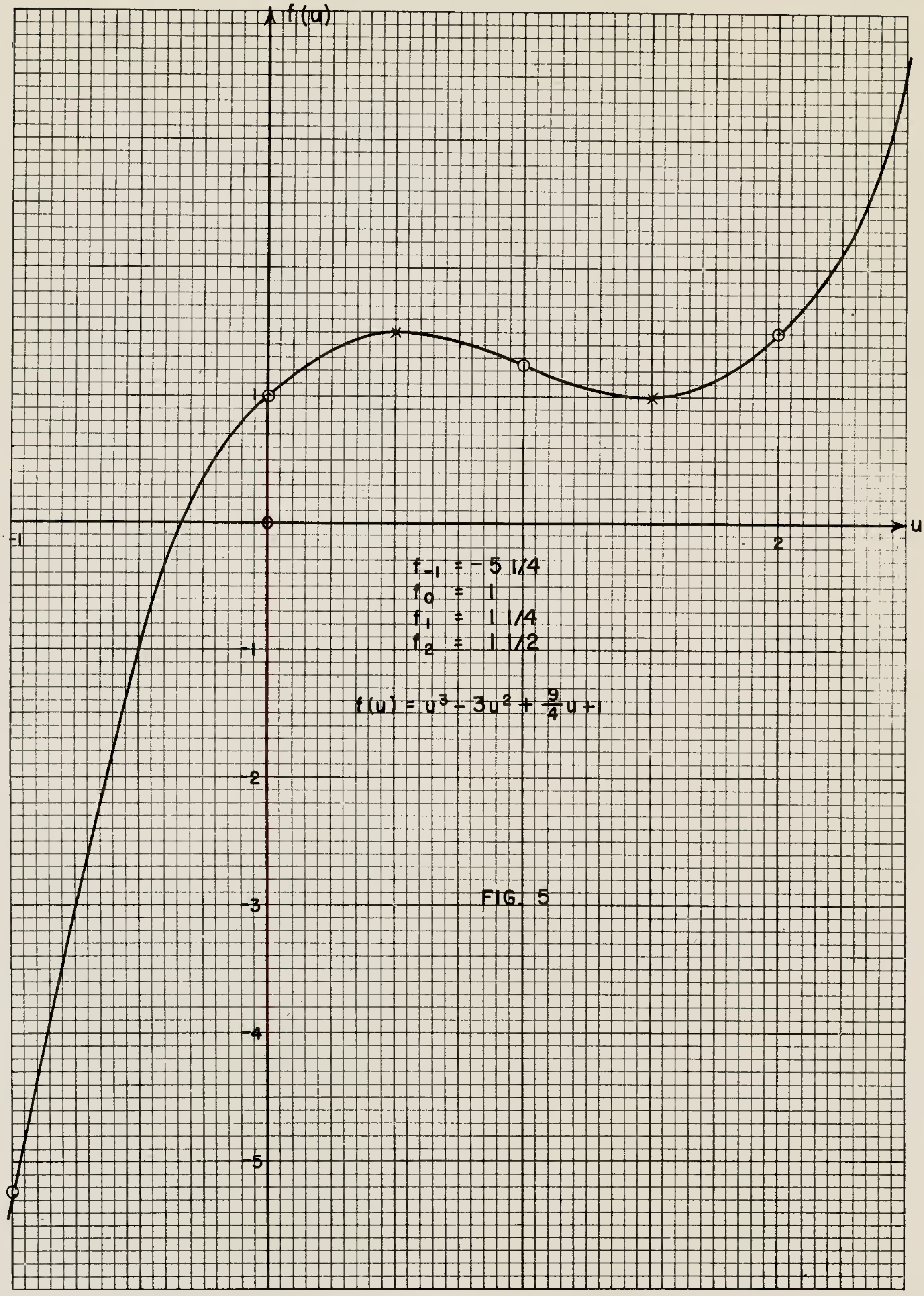


FIG. 5

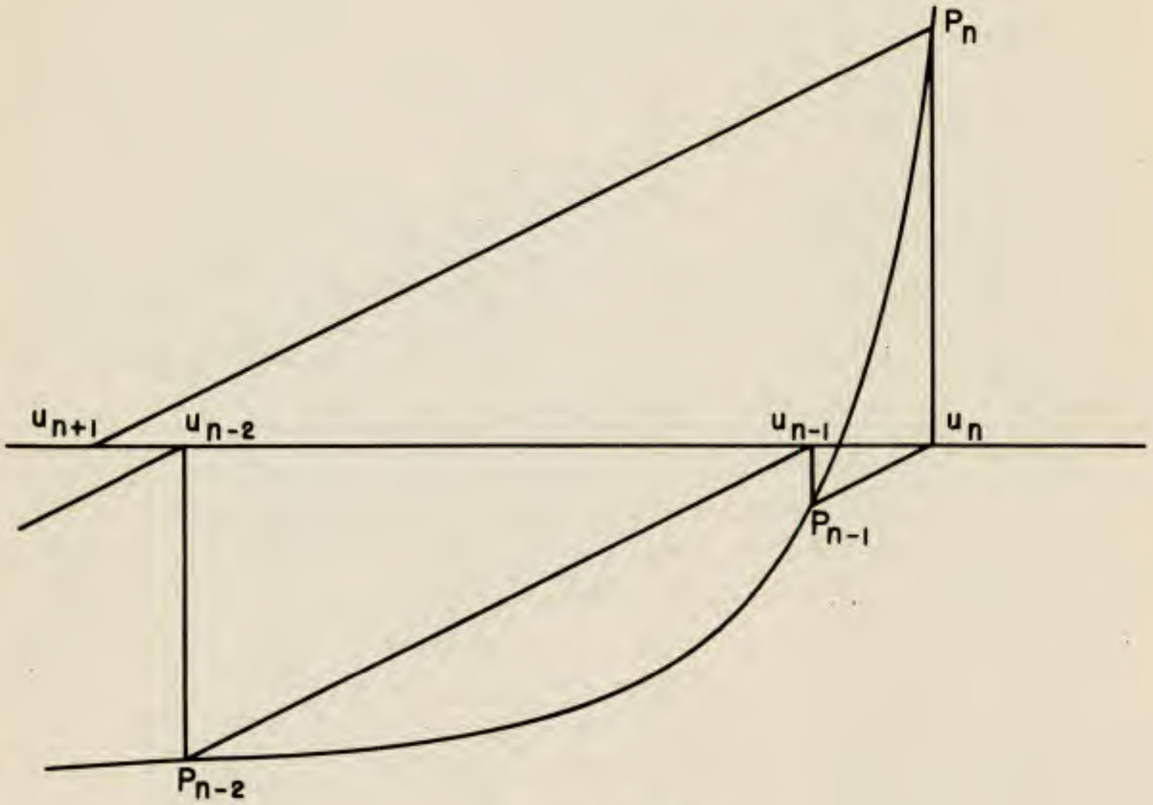


FIG. 6

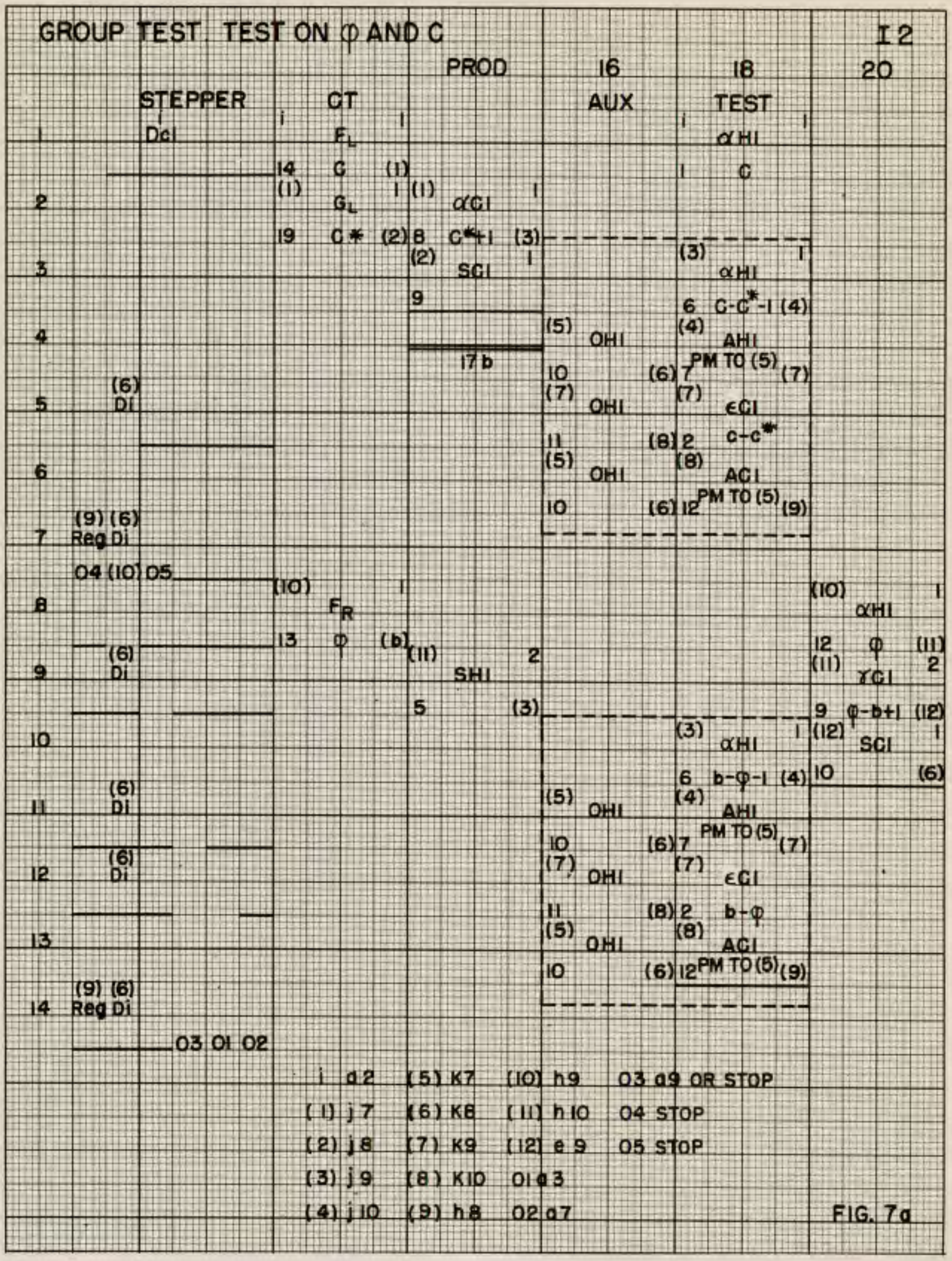

GROUP TEST. TEST ON φ AND C
I2
STEPPER
CT
PROD
16
AUX
18
TEST
20
i Dci
i F_L 1
i αHI 1
14 C (1)
1 C
(1) G_L 1
(1) αCI 1
19 C* (2)
8 C*+1 (3)
(2) SCI 1
(3) αHI 1
9
6 C-C*-1 (4)
(4) AHI
(5) OHI
17b
10 (6) 7 PM TO (5) (7)
(6) Di
(7) OHI
(7) ϵCI
11 (8) 2 c-c*
(5) OHI
(8) ACI
10 (6) 12 PM TO (5) (9)
(9) (6) Reg Di
04 (10) 05
(10) F_R 1
(10) αHI 1
13 φ (b)
12 φ (11)
(11) SHI 2
(11) γCI 2
(6) Di
5 (3)
9 φ-b+1 (12)
(3) αHI 1
(12) SCI 1
10 (6)
6 b-φ-1 (4)
(6) Di
(5) OHI
(4) AHI
10 (6) 7 PM TO (5) (7)
(6) Di
(7) OHI
(7) ϵCI
11 (8) 2 b-φ
(5) OHI
(8) ACI
10 (6) 12 PM TO (5) (9)
(9) (6) Reg Di
03 01 02
1 2 3 4 5 6 7 8 9 10 11 12 13 14
i d2
(1) j7
(2) j8
(3) j9
(4) j10
(5) K7
(6) K8
(7) K9
(8) K10
(9) h8
(10) h9
(11) h10
(12) e9
01 a3
02 a7
03 a9 OR STOP
04 STOP
05 STOP

FIG. 7a

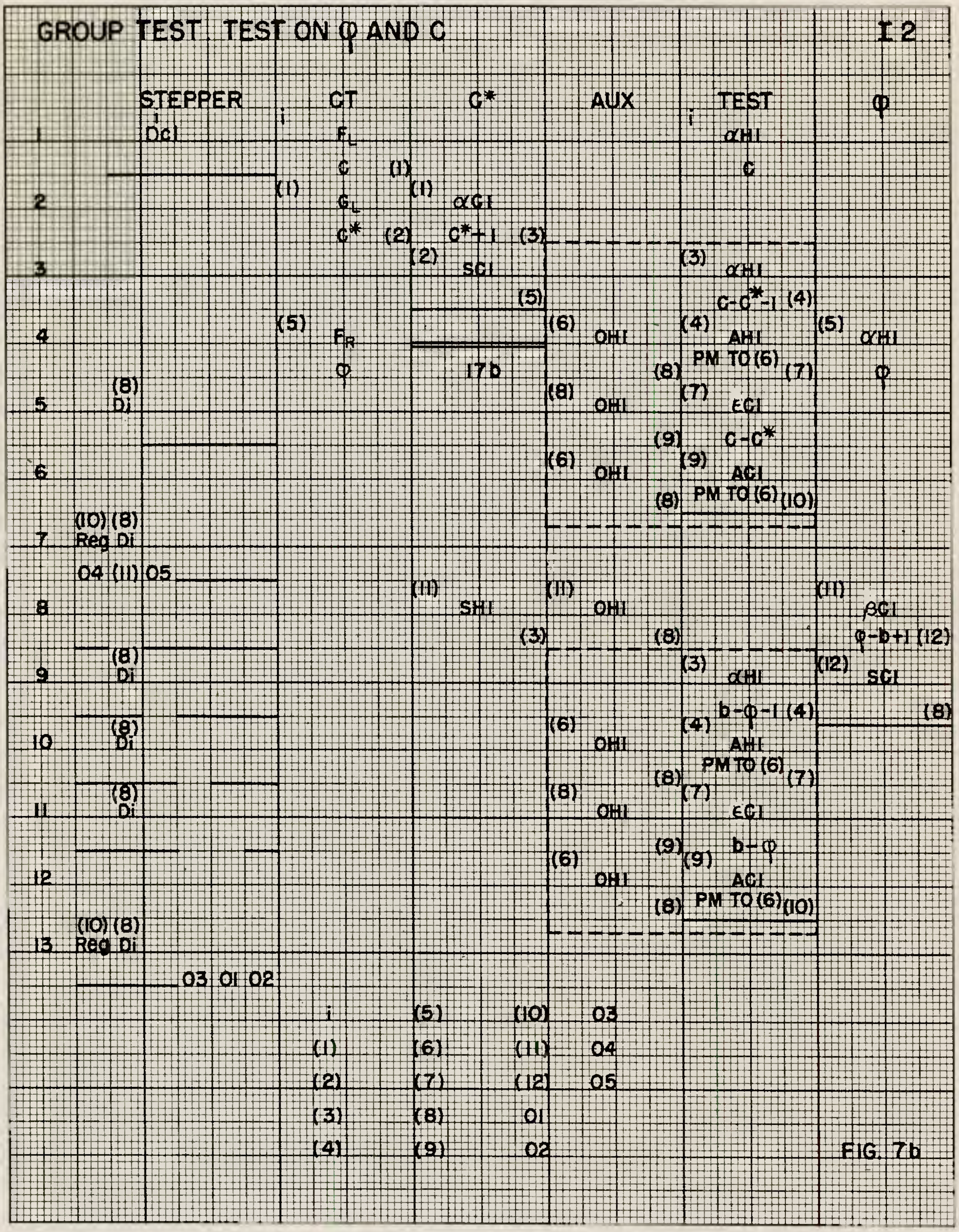

GROUP TEST. TEST ON φ AND C
I 2
STEPPER
CT
C*
AUX
TEST
φ
1
2
3
4
5
6
7
8
9
10
11
12
13
i
Dcl
F_L
C
G_L
C*
F_R
φ
(1)
(2)
(1)
(2)
(3)
(5)
αCI
C*+I
SCI
17b
αHI
C
(3)
αHI
C-C*-I (4)
(4)
AHI
PM TO (6)
(7)
(7)
εCI
(9)
C-C*
(9)
ACI
PM TO (6) (10)
(5)
αHI
φ
(6)
OHI
(8)
OHI
(6)
OHI
(8)
(8)
(8)
Di
(10) (8)
Reg Di
O4 (11) O5
(11)
SHI
(11)
OHI
(3)
(8)
(11)
βCI
φ-b+I (12)
(8)
Di
(3)
αHI
(12)
SCI
(8)
Di
(6)
OHI
(4)
b-φ-I (4)
AHI
PM TO (6)
(7)
(8)
(8)
Di
(8)
OHI
(7)
εCI
(6)
OHI
(9)
(9)
b-φ
ACI
(8)
PM TO (6) (10)
(10) (8)
Reg Di
O3 O1 O2
i
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)
O1
O2
O3
O4
O5

FIG. 7b

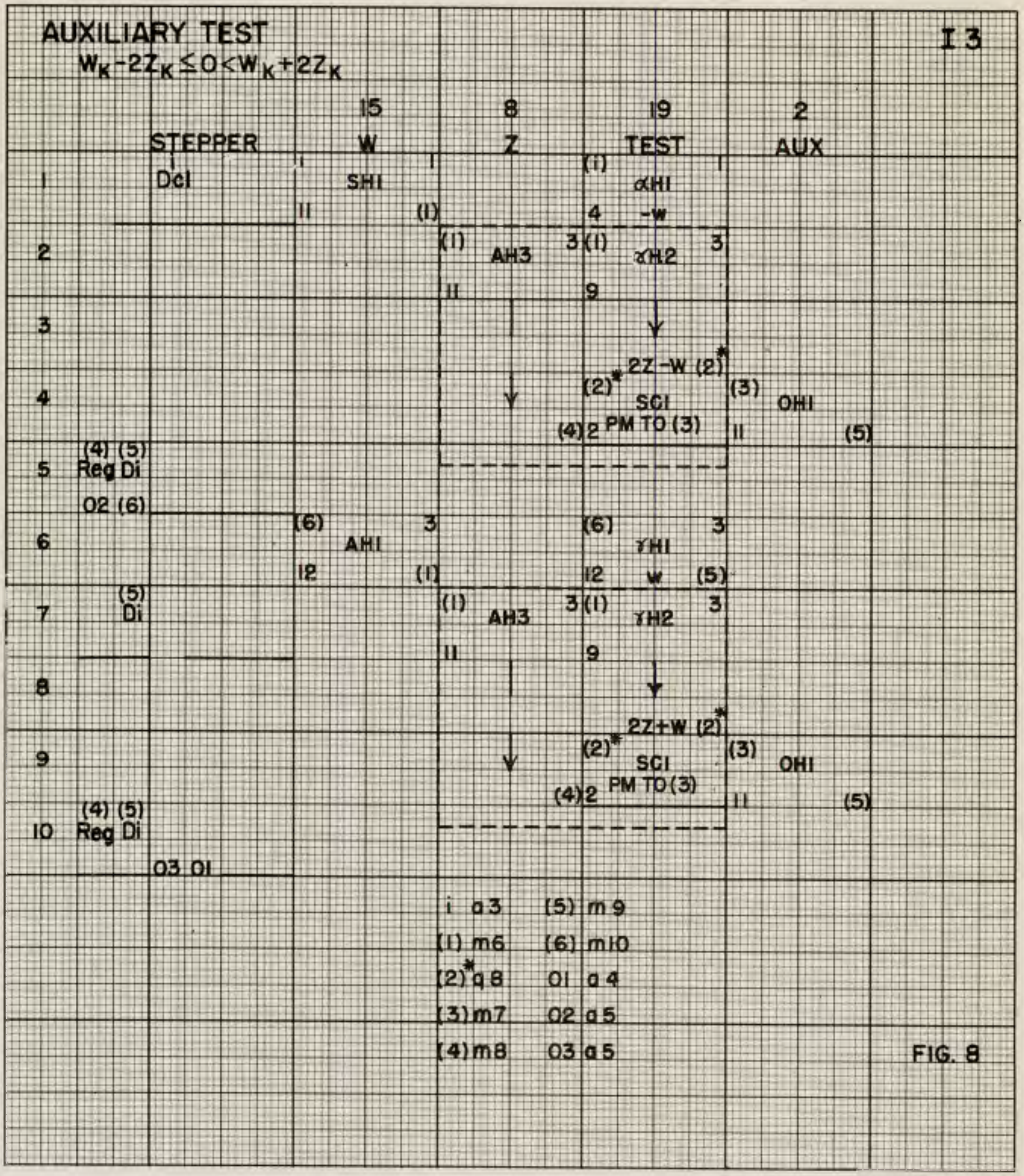

AUXILIARY TEST
$W_K - 2Z_K \leq 0 < W_K + 2Z_K$
I 3
15 W
8 Z
19 TEST
2 AUX
STEPPER
1 Dcl
SHI
(1) αHI
4 -w
11 (1)
2 (1) AH3 3
(1) γH2 3
11
9
3
2Z -W (2)*
4 (2)* SCI
(3) OHI
(4) 2 PM TO (3)
11 (5)
(4) (5)
5 Reg Di
O2 (6)
6 (6) AHI 3
(6) γHI 3
12 (1)
12 w (5)
7 (5) Di
(1) AH3 3
(1) γH2 3
11
9
8
2Z+W (2)*
9 (2)* SCI
(3) OHI
(4) 2 PM TO (3)
11 (5)
(4) (5)
10 Reg Di
O3 OI
i a3
(5) m9
(1) m6
(6) m10
(2)* a8
O1 a4
(3) m7
O2 a5
(4) m8
O3 a5

FIG. 8

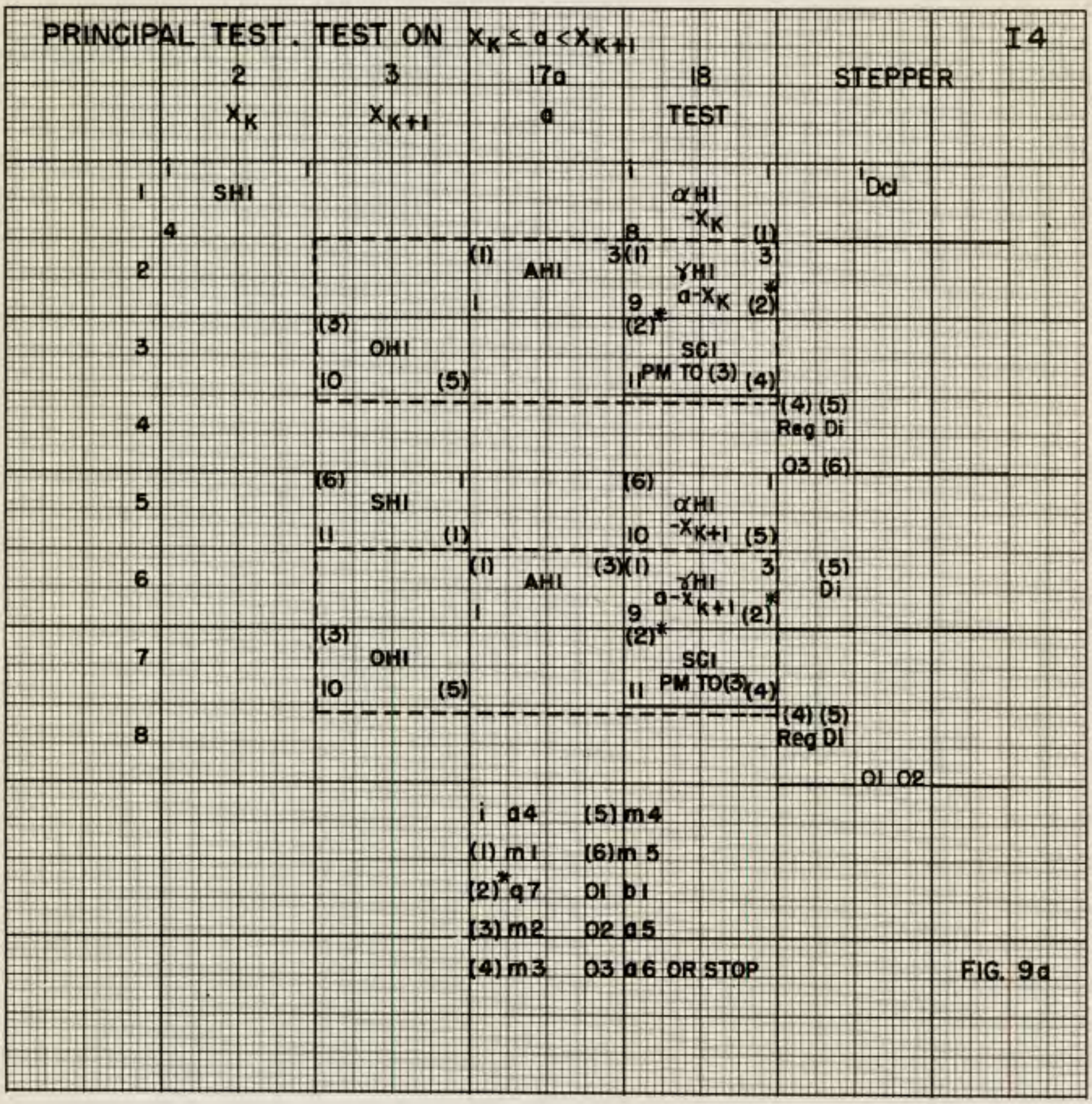
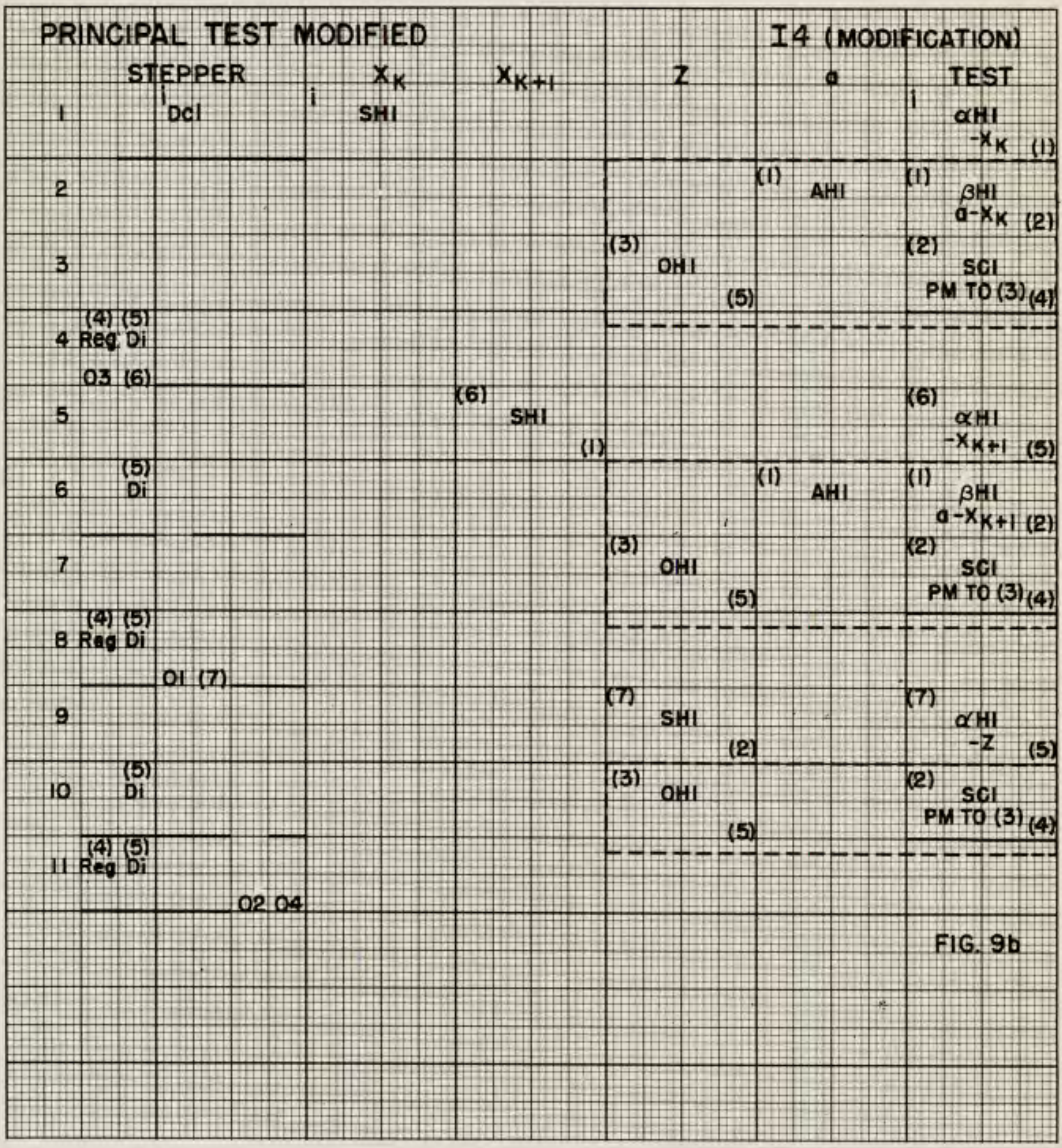

PRINCIPAL TEST. TEST ON $x_K \leq a < x_{K+1}$
I4
2
x_K
3
x_{K+1}
17a
a
18
TEST
STEPPER
1
SHI
4
αHI
$-x_K$
Dd
(1) AHI 3 (1)
γHI
$a-x_K$
(2)*
(3)
OHI
10 (5)
SCI
PM TO (3)
(4)
(4) (5)
Reg Di
O3 (6)
(6) SHI
αHI
$-x_{K+1}$
11 (1)
10 (5)
(1) AHI (3)(1)
γHI
$a-x_{K+1}$
(2)*
(5)
Di
(3)
OHI
10 (5)
SCI
PM TO (3)
(4)
(4) (5)
Reg DI
O1 O2
1
2
3
4
5
6
7
8
i a4
(1) m1
(2)* q7
(3) m2
(4) m3
(5) m4
(6) m5
O1 b1
O2 a5
O3 a6 OR STOP

FIG. 9a



PRINCIPAL TEST MODIFIED

I4 (MODIFICATION)

	STEPPER	X_K	X_{K+1}	Z	a	TEST
1	i DCI	i SHI				i αHI $-X_K$ (1)
2					(1) AHI	(1) βHI $a-X_K$ (2)
3				(3) OHI (5)		(2) SCI PM TO (3) (4)
4	(4) (5) Reg Di 03 (6)					
5			(6) SHI (1)			(6) αHI $-X_{K+1}$ (5)
6	(5) Di				(1) AHI	(1) βHI $a-X_{K+1}$ (2)
7				(3) OHI (5)		(2) SCI PM TO (3) (4)
8	(4) (5) Reg Di OI (7)					
9				(7) SHI (2)		(7) αHI $-Z$ (5)
10	(5) Di			(3) OHI (5)		(2) SCI PM TO (3) (4)
11	(4) (5) Reg Di 02 04					

FIG. 9b

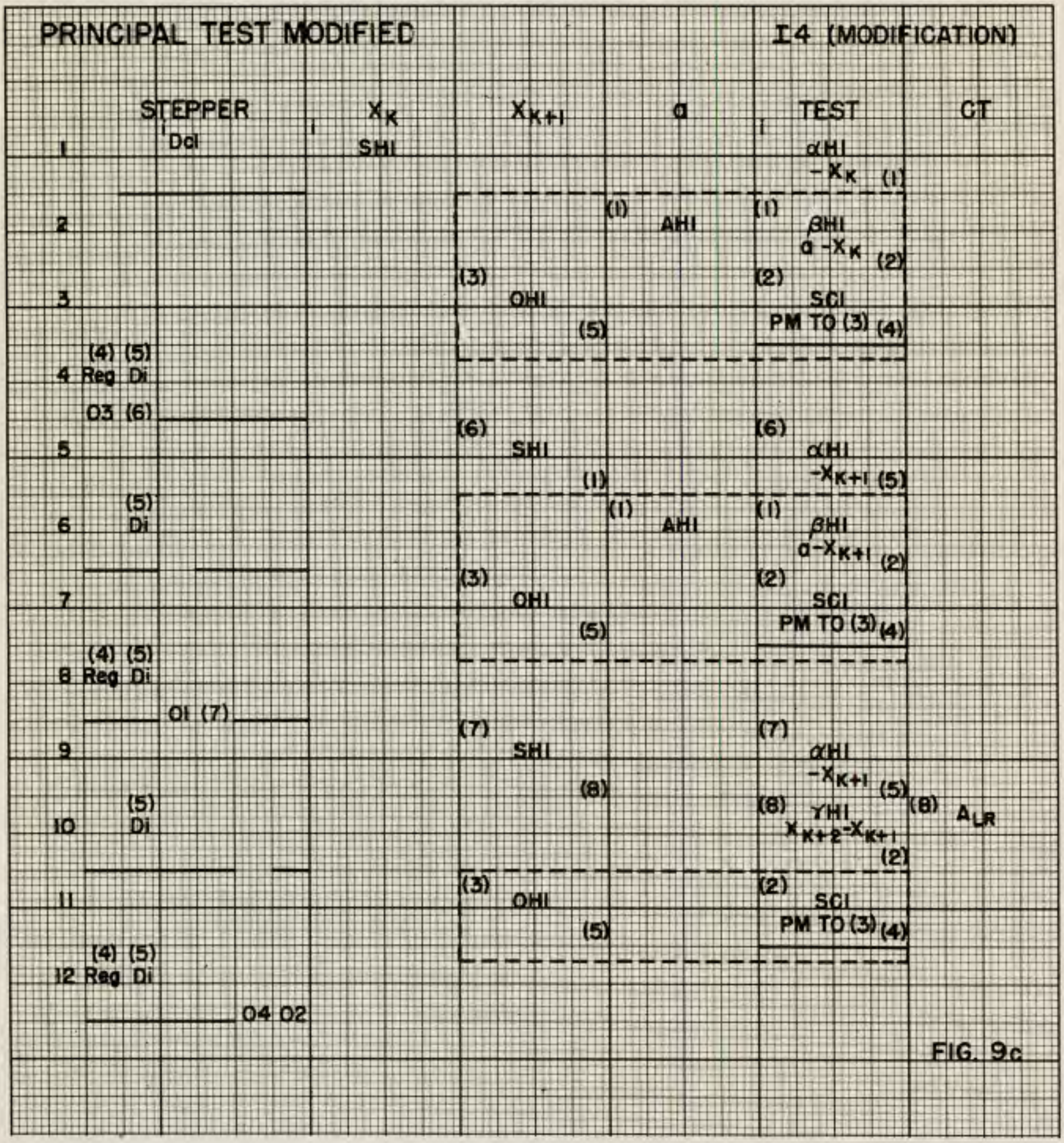
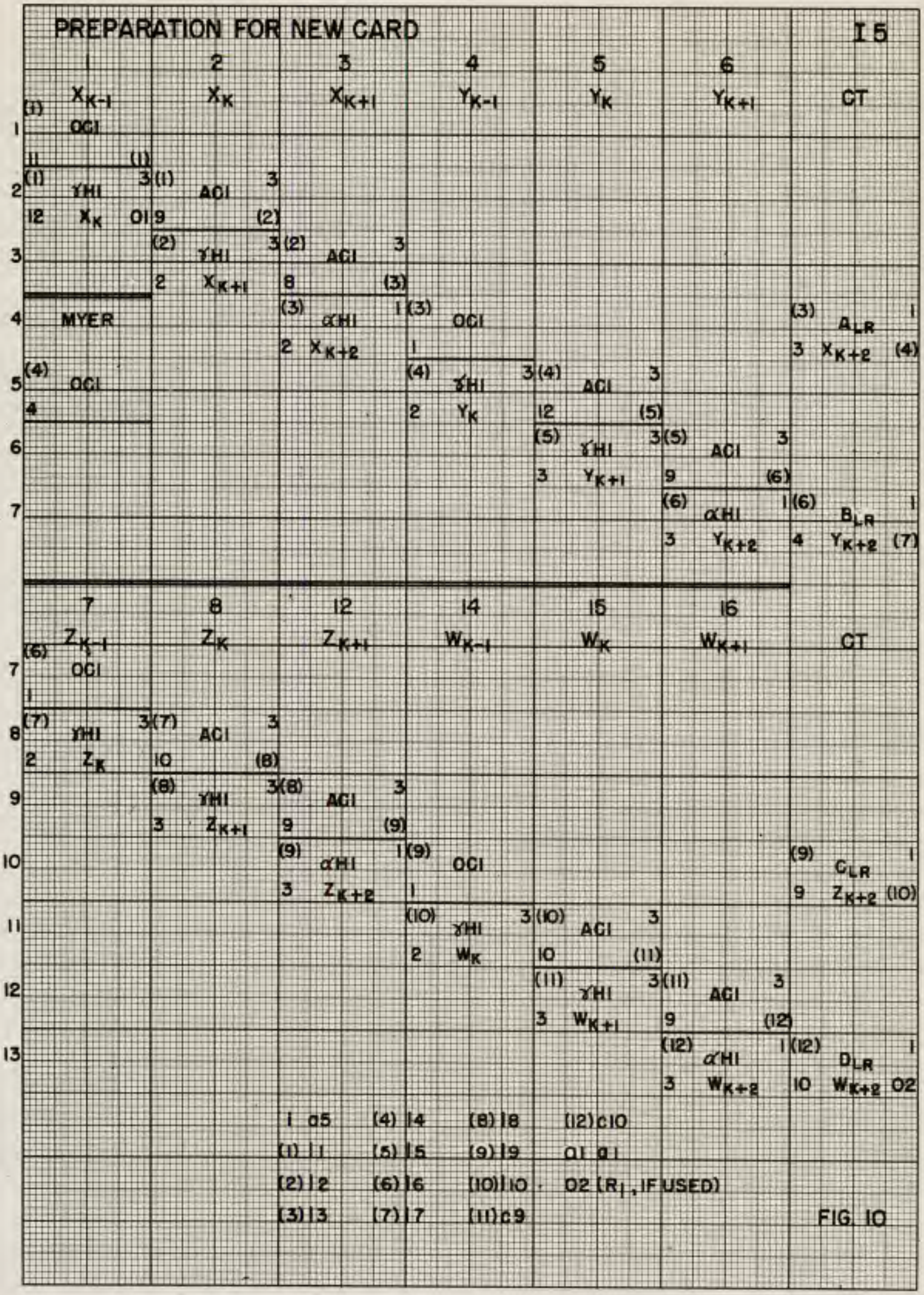

PRINCIPAL TEST MODIFIED
I4 (MODIFICATION)
STEPPER
X_K
X_{K+1}
a
TEST
CT
1
Dci
SHI
αHI
$-X_K$ (1)
2
(1)
AHI
(1)
βHI
$a-X_K$ (2)
(3)
(2)
3
OHI
SCI
(5)
PM TO (3) (4)
(4) (5)
4 Reg Di
O3 (6)
(6)
(6)
5
SHI
αHI
$-X_{K+1}$ (5)
(1)
(5)
6 Di
(1)
AHI
(1)
βHI
$a-X_{K+1}$ (2)
(3)
(2)
7
OHI
SCI
(5)
PM TO (3) (4)
(4) (5)
8 Reg Di
O1 (7)
(7)
(7)
9
SHI
αHI
$-X_{K+1}$ (5)
(8)
(5)
10 Di
(8)
γHI
$X_{K+2}-X_{K+1}$
(8)
A_{LR}
(2)
11
(3)
OHI
(2)
SCI
(5)
PM TO (3) (4)
(4) (5)
12 Reg Di
O4 O2

FIG. 9c

PREPARATION FOR NEW CARD

I5



1 a5	(4) 14	(8) 18	(12) c10
(1) 11	(5) 15	(9) 19	01 a1
(2) 12	(6) 16	(10) 110	02 (R_1, IF USED)
(3) 13	(7) 17	(11) c9	

FIG. 10

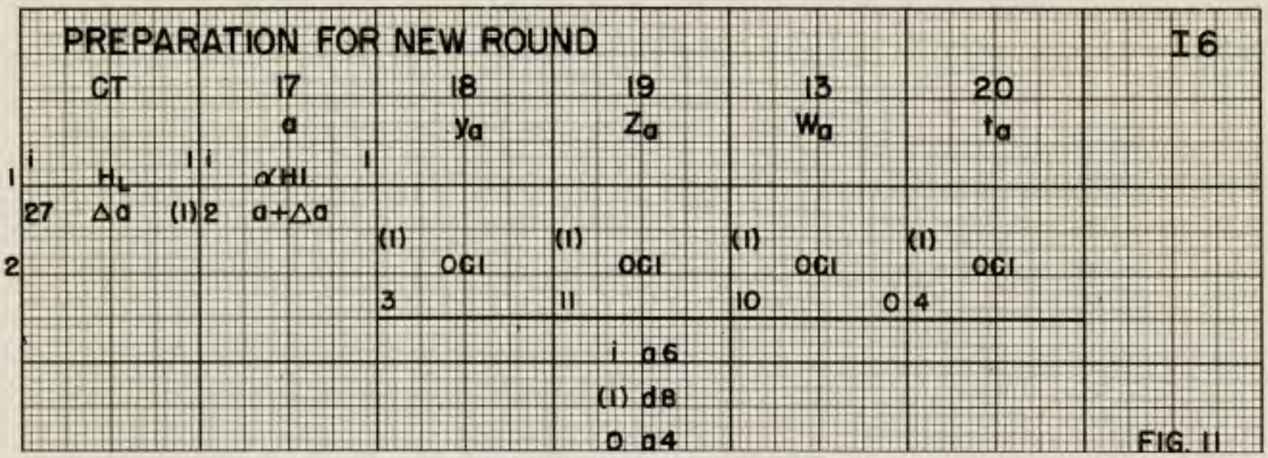


FIG. 11

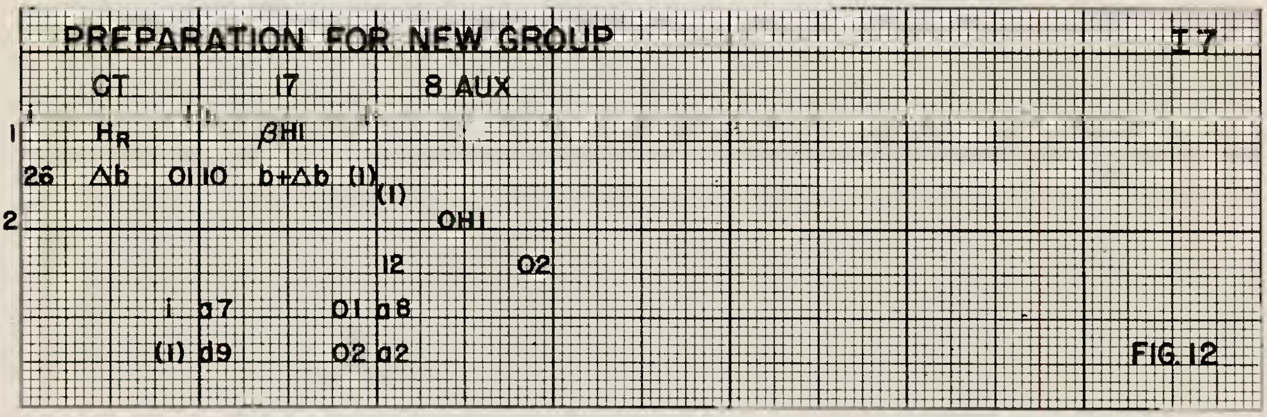


FIG. 12

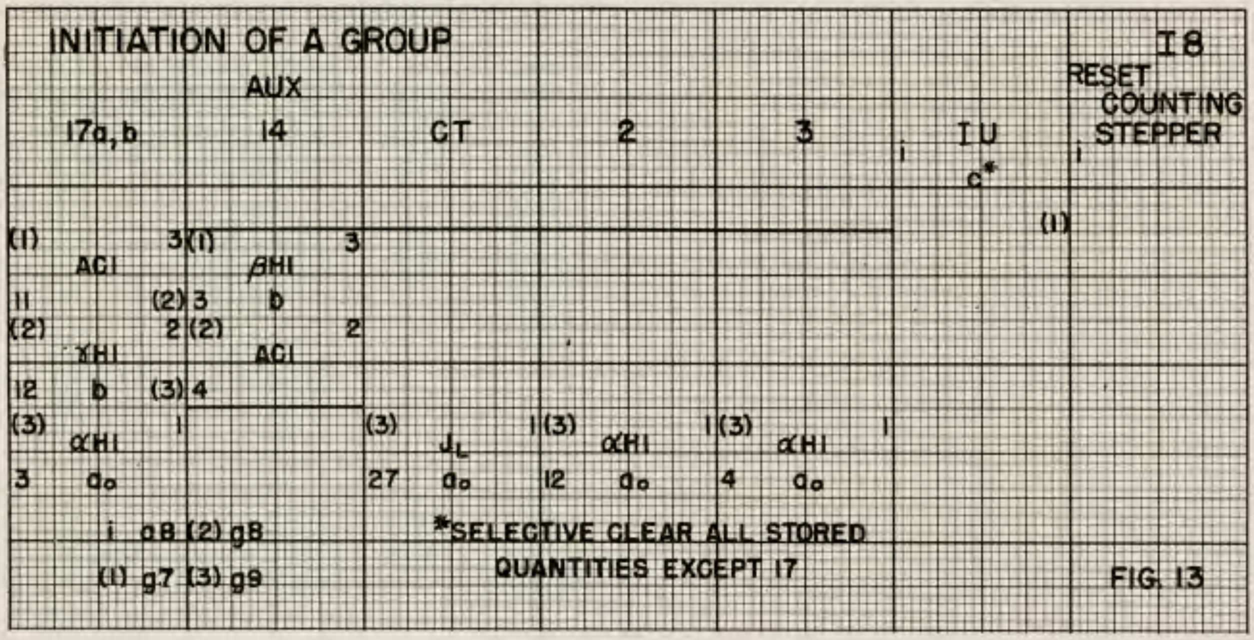


FIG. 13

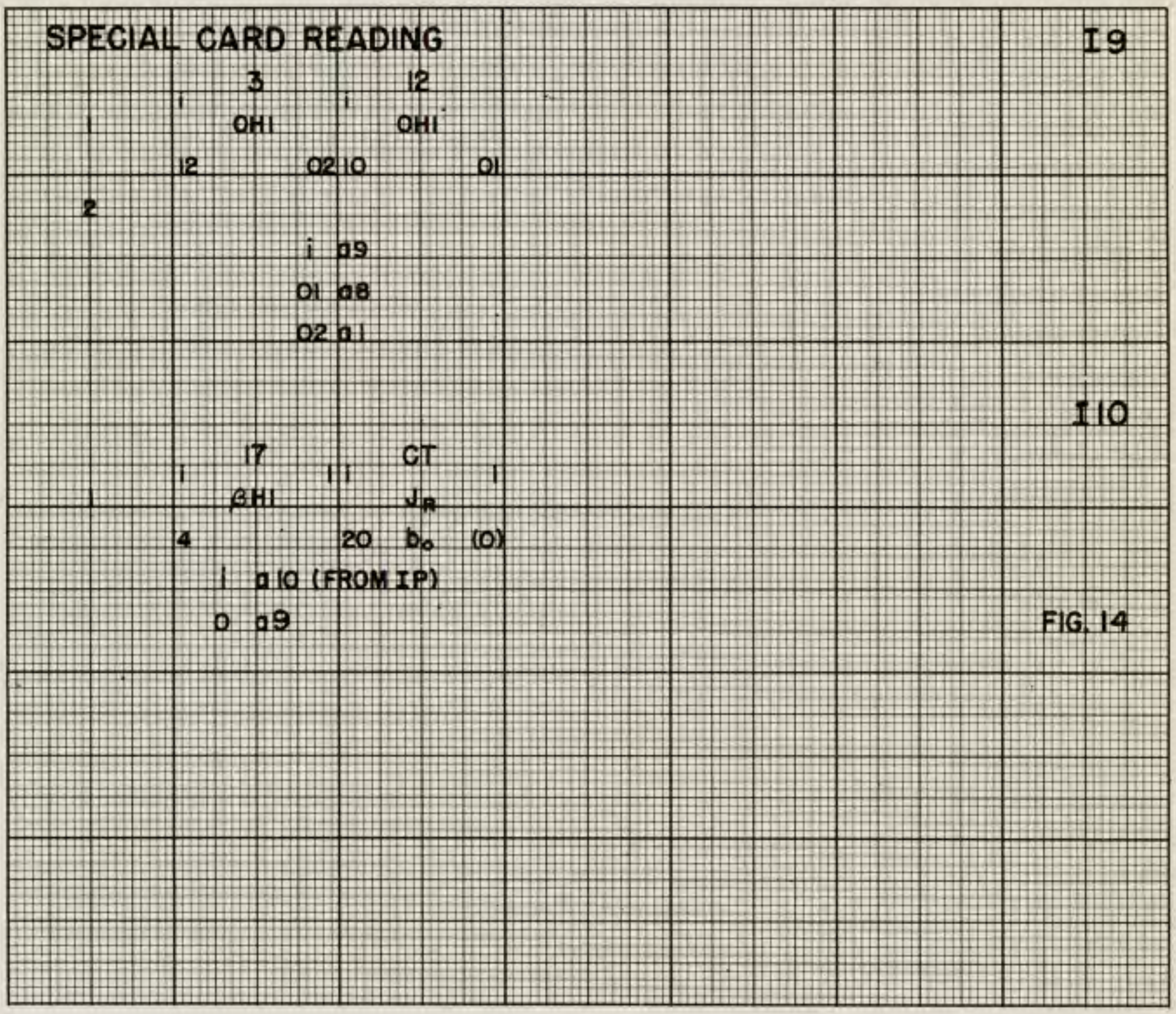

SPECIAL CARD READING
I9
3
12
1
i
i
OHI
OHI
12
02
10
01
2
i
a9
01
a8
02
a1
I10
17
CT
1
1
1 1
1
βHI
J_R
4
20
b_o
(0)
1
a10 (FROM IP)
0
a9

FIG. 14

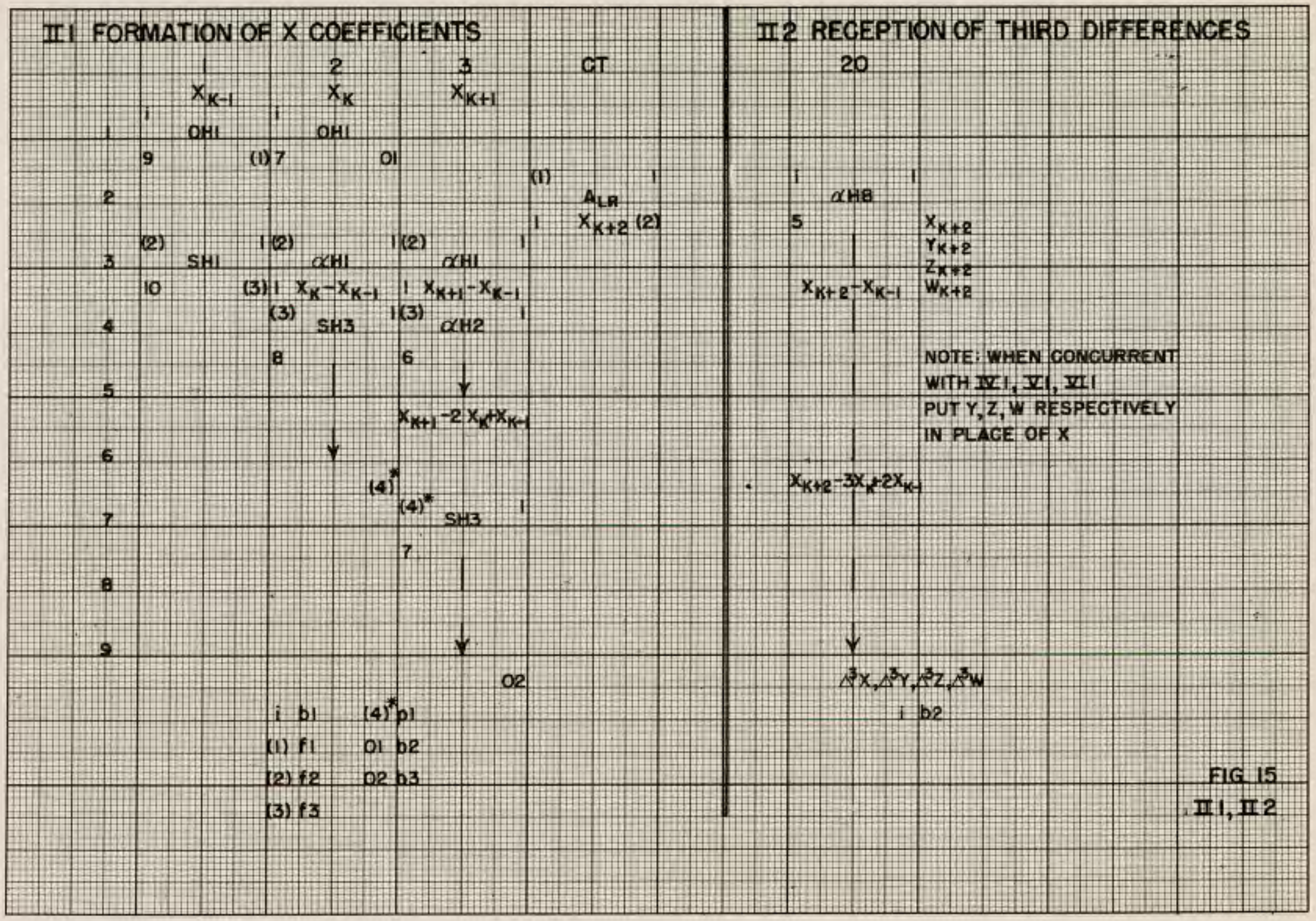
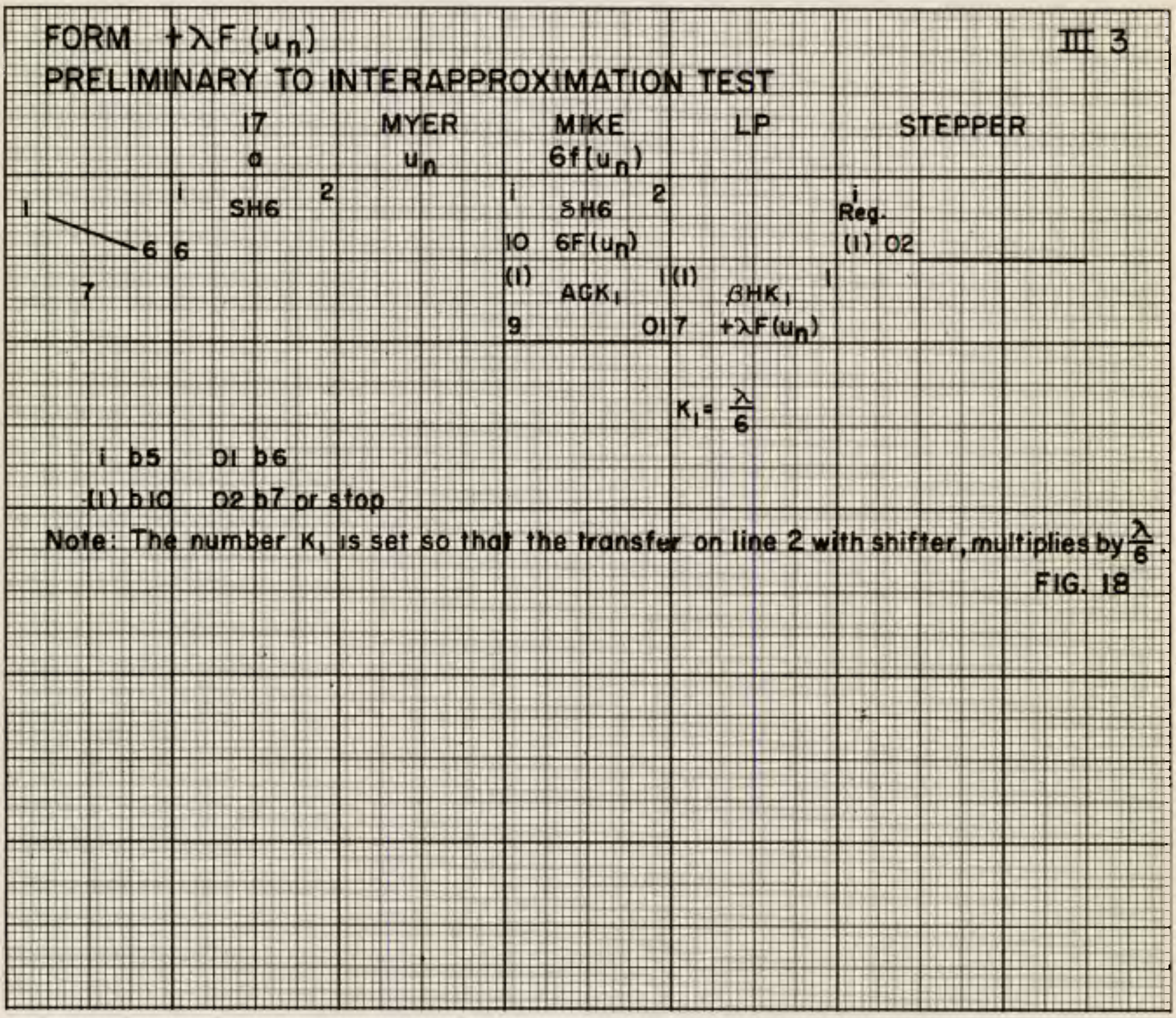


FIG 15
II1, II2



FORM $+\lambda F(u_n)$ III 3

PRELIMINARY TO INTERAPPROXIMATION TEST

	17	MYER	MIKE	LP	STEPPER
	α	u_n	$6f(u_n)$		
1	i SH6 2		i δH6 2		i Reg.
6	6		10 $6F(u_n)$		(1) 02
7			(1) ACK_1 1	(1) βHK_1 1	
			9 01	7 $+\lambda F(u_n)$	

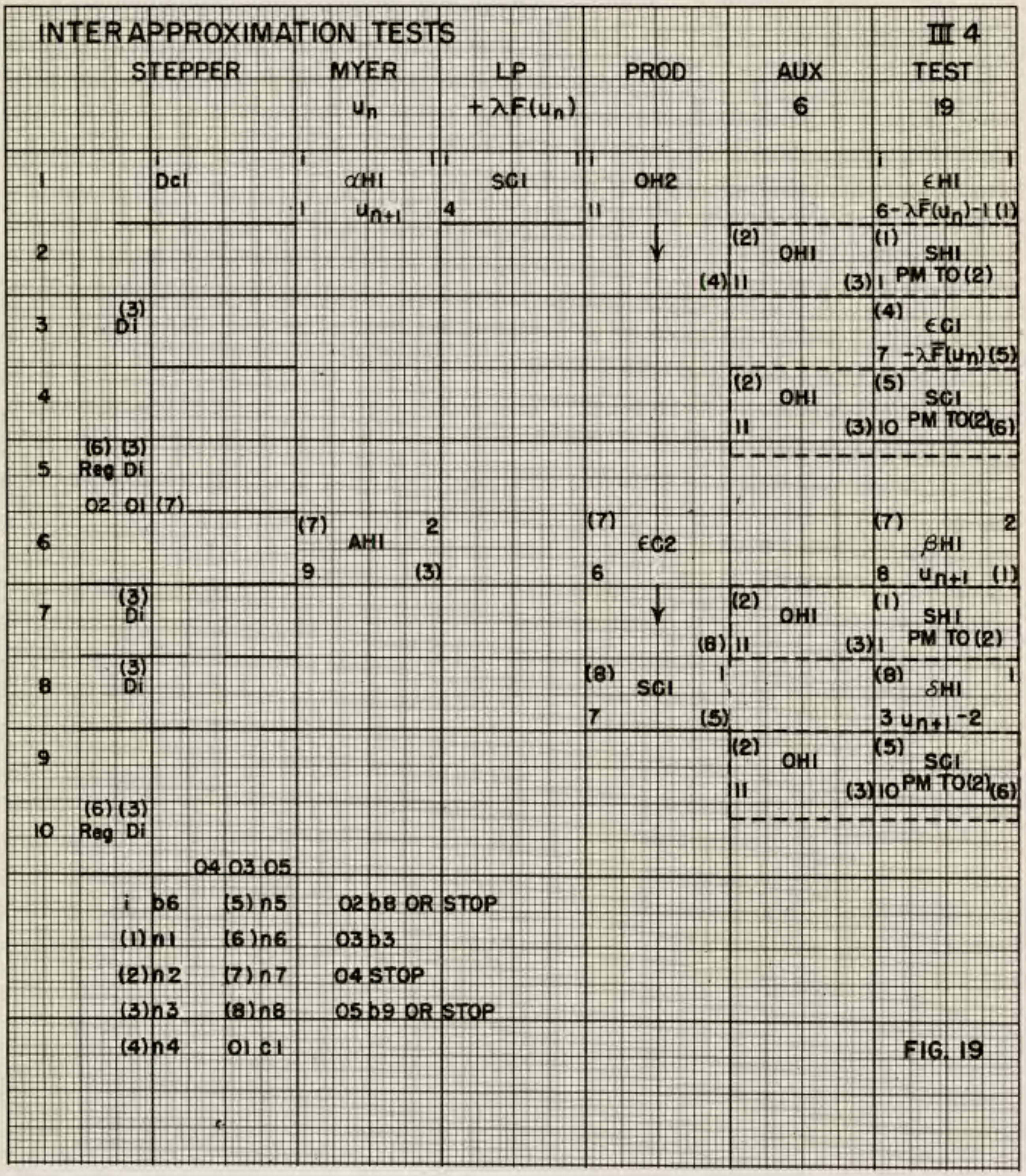
$K_1 = \frac{\lambda}{6}$

i b5 01 b6

(1) b10 02 b7 or stop

Note: The number K_1 is set so that the transfer on line 2 with shifter, multiplies by $\frac{\lambda}{6}$.

FIG. 18



INTERAPPROXIMATION TESTS III 4

	STEPPER	MYER	LP	PROD	AUX	TEST
		u_n	$+\lambda F(u_n)$		6	19
1	i Dcl	i 1 αHI 1 u_{n+1}	i 1 SCI 4	i OH2 11		i 1 εHI 6 $-\lambda\bar{F}(u_n)-1$ (1)
2				↓ (4)	(2) OHI 11 (3)	(1) SHI 1 PM TO (2)
3	(3) Di					(4) εCI 7 $-\lambda\bar{F}(u_n)$ (5)
4					(2) OHI 11 (3)	(5) SCI 10 PM TO(2) (6)
5	(6) (3) Reg Di 02 01 (7)					
6		(7) 2 AHI 9 (3)		(7) εC2 6		(7) 2 βHI 8 u_{n+1} (1)
7	(3) Di			↓ (8)	(2) OHI 11 (3)	(1) SHI 1 PM TO (2)
8	(3) Di			(8) 1 SCI 7 (5)		(8) 1 δHI 3 u_{n+1}^{-2}
9					(2) OHI 11 (3)	(5) SCI 10 PM TO(2) (6)
10	(6) (3) Reg Di 04 03 05					

i b6	(5) n5	02 b8 OR STOP
(1) n1	(6) n6	03 b3
(2) n2	(7) n7	04 STOP
(3) n3	(8) n8	05 b9 OR STOP
(4) n4	01 c1	

FIG. 19

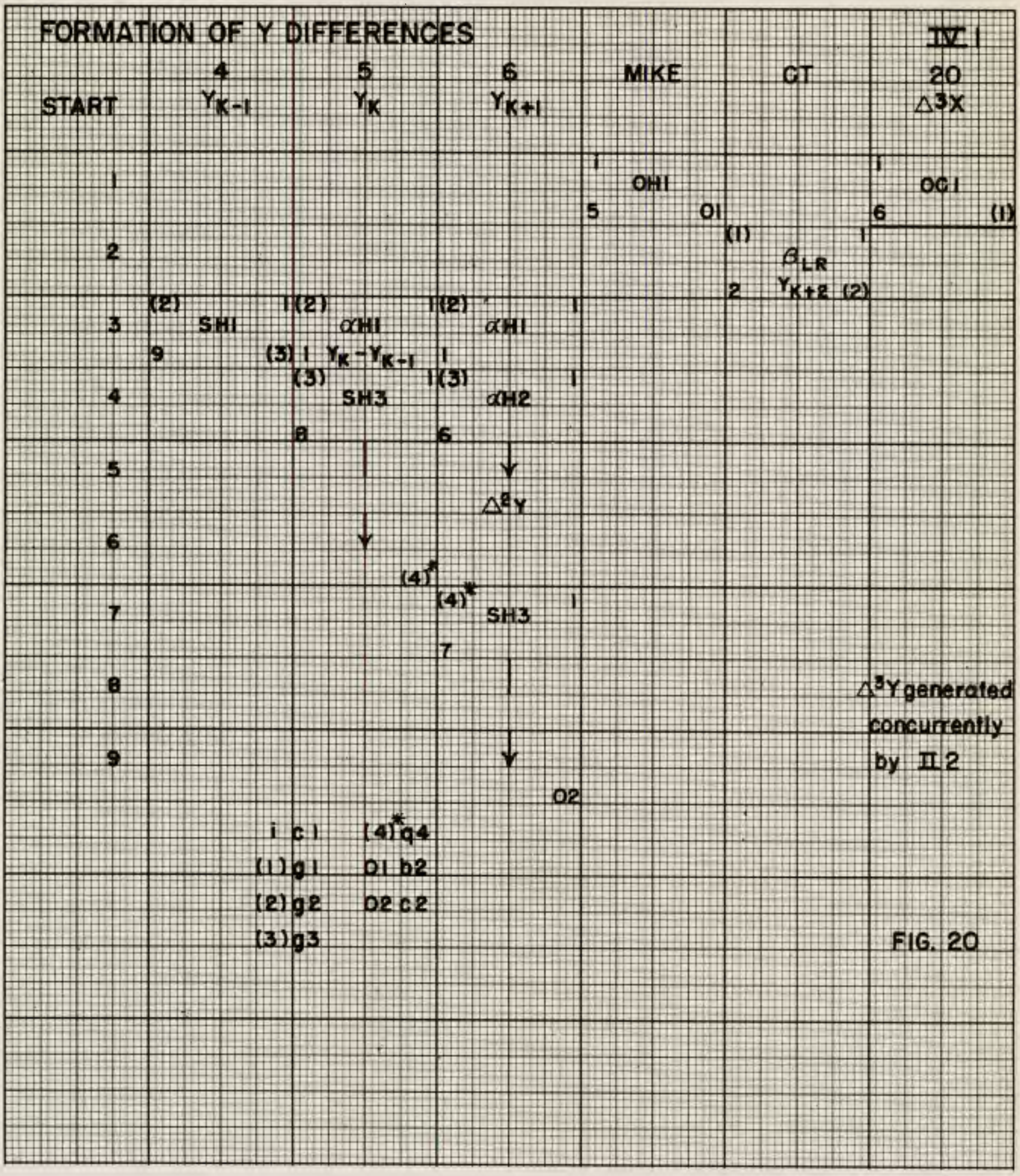

FORMATION OF Y DIFFERENCES
IV 1
START
4
Y_{K-1}
5
Y_K
6
Y_{K+1}
MIKE
CT
20
Δ^3X
1
2
3
4
5
6
7
8
9
OH1
O1
OC1
(1)
β_{LR}
Y_{K+2} (2)
(2)
SH1
αH1
(3)
$Y_K - Y_{K-1}$
SH3
αH2
Δ^2Y
(4)*
SH3
O2
Δ^3Y generated concurrently by II 2
i c1
(1) g1
(2) g2
(3) g3
(4)* g4
O1 b2
O2 c2

FIG. 20

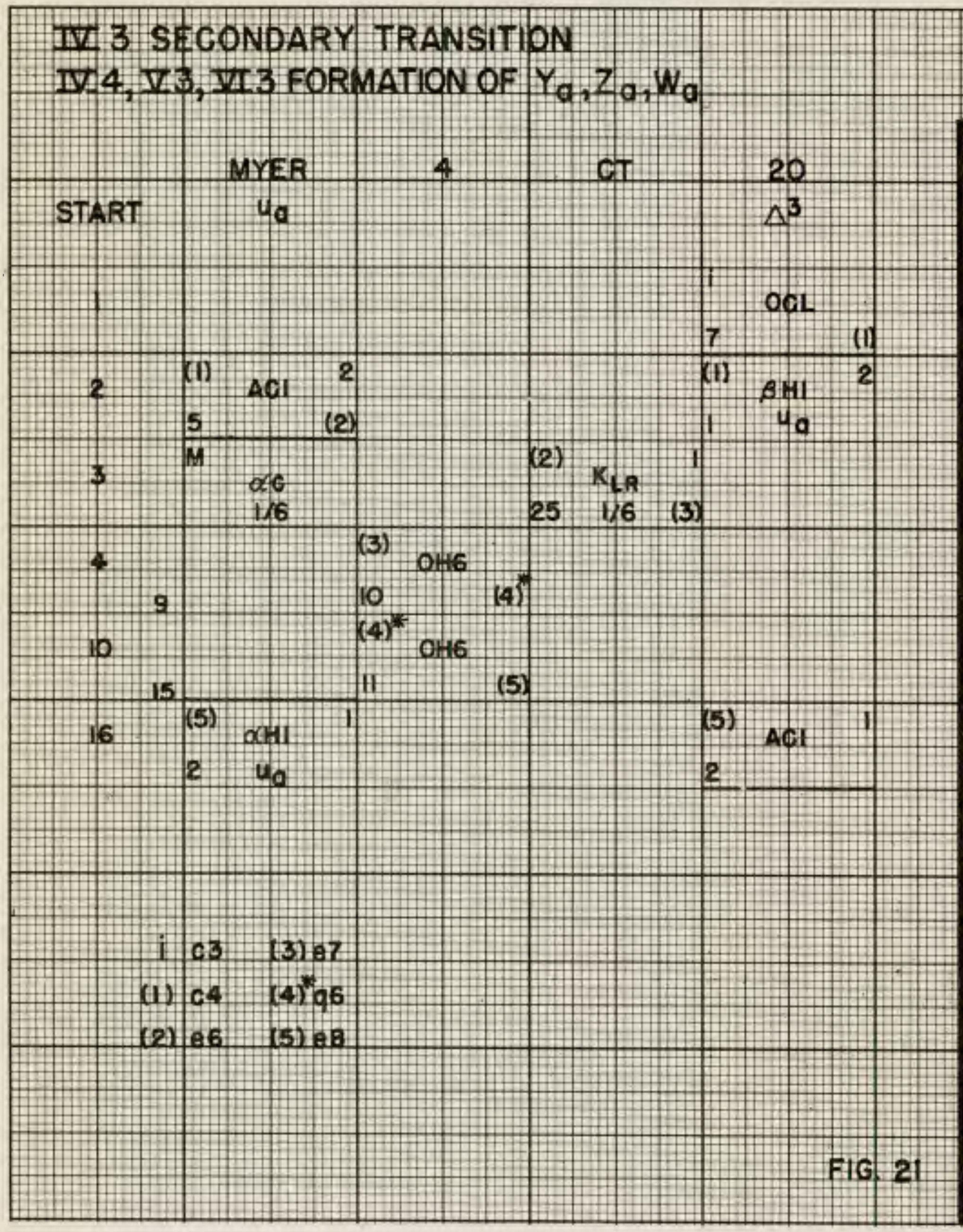

IV 3 SECONDARY TRANSITION
IV 4, V 3, VI 3 FORMATION OF Y_a, Z_a, W_a
MYER
4
CT
20
START
u_a
Δ^3
1
i
OCL
7
(1)
2
(1)
ACI
2
(1)
βHI
2
5
(2)
1
u_a
3
M
αC
1/6
(2)
K_{LR}
25
1/6
(3)
1
4
(3)
OH6
9
10
(4)*
10
(4)*
OH6
15
11
(5)
16
(5)
αHI
1
2
u_a
(5)
ACI
1
2
i c3 (3) e7
(1) c4 (4)* a6
(2) e6 (5) e8

FIG. 21

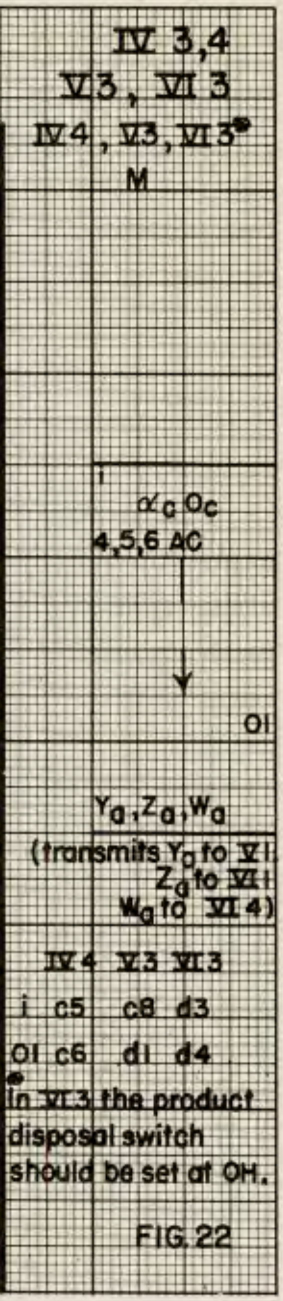
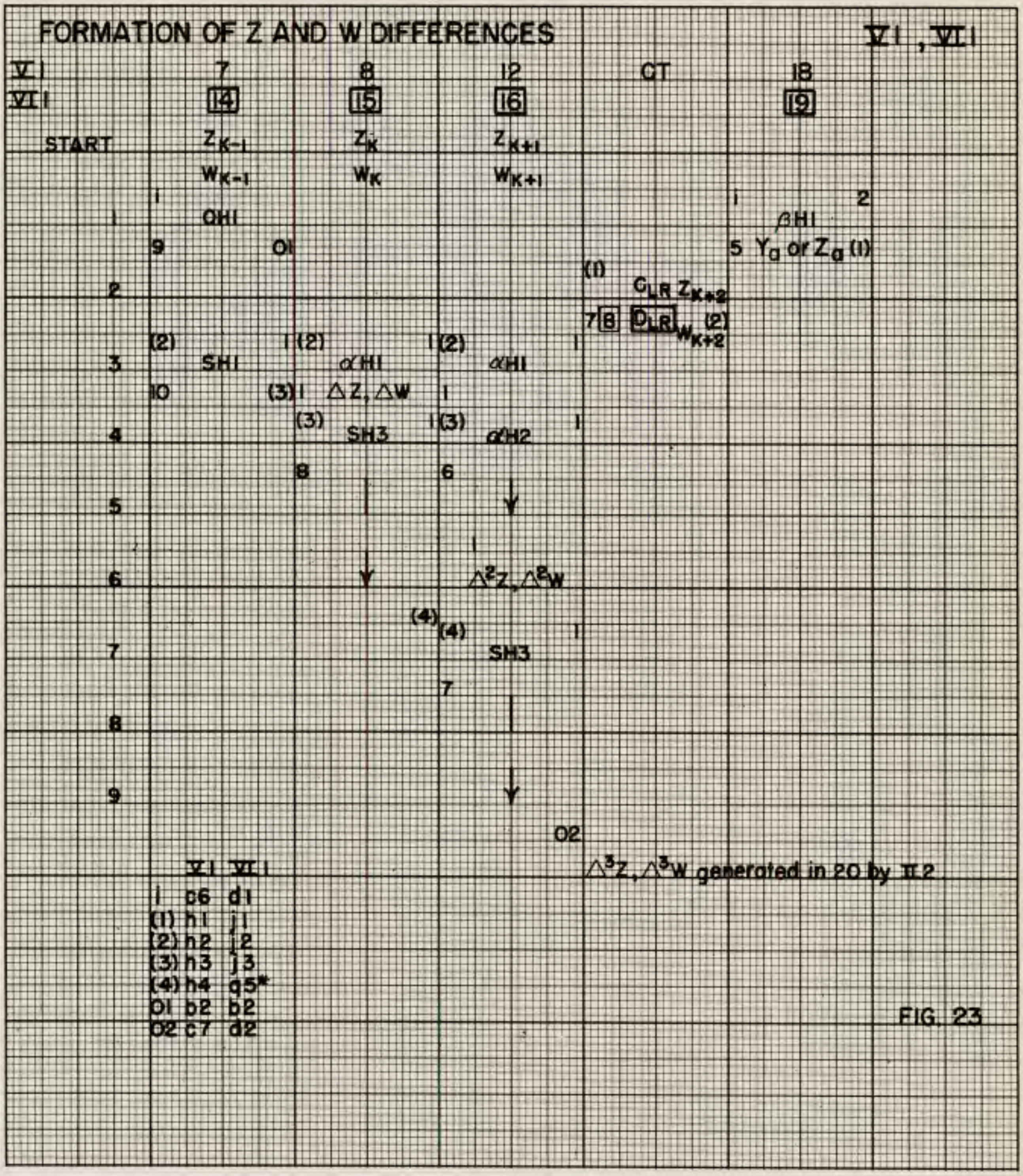

IV 3,4
V 3, VI 3
IV 4, V 3, VI 3*
M
1
α_C 0_C
4,5,6 AC
OI
Y_a, Z_a, W_a
(transmits Y_a to V 1
Z_a to VI 1
W_a to VI 4)
IV 4 V 3 VI 3
i c5 c8 d3
OI c6 d1 d4
* In VI 3 the product disposal switch should be set at OH.

FIG. 22



FORMATION OF Z AND W DIFFERENCES

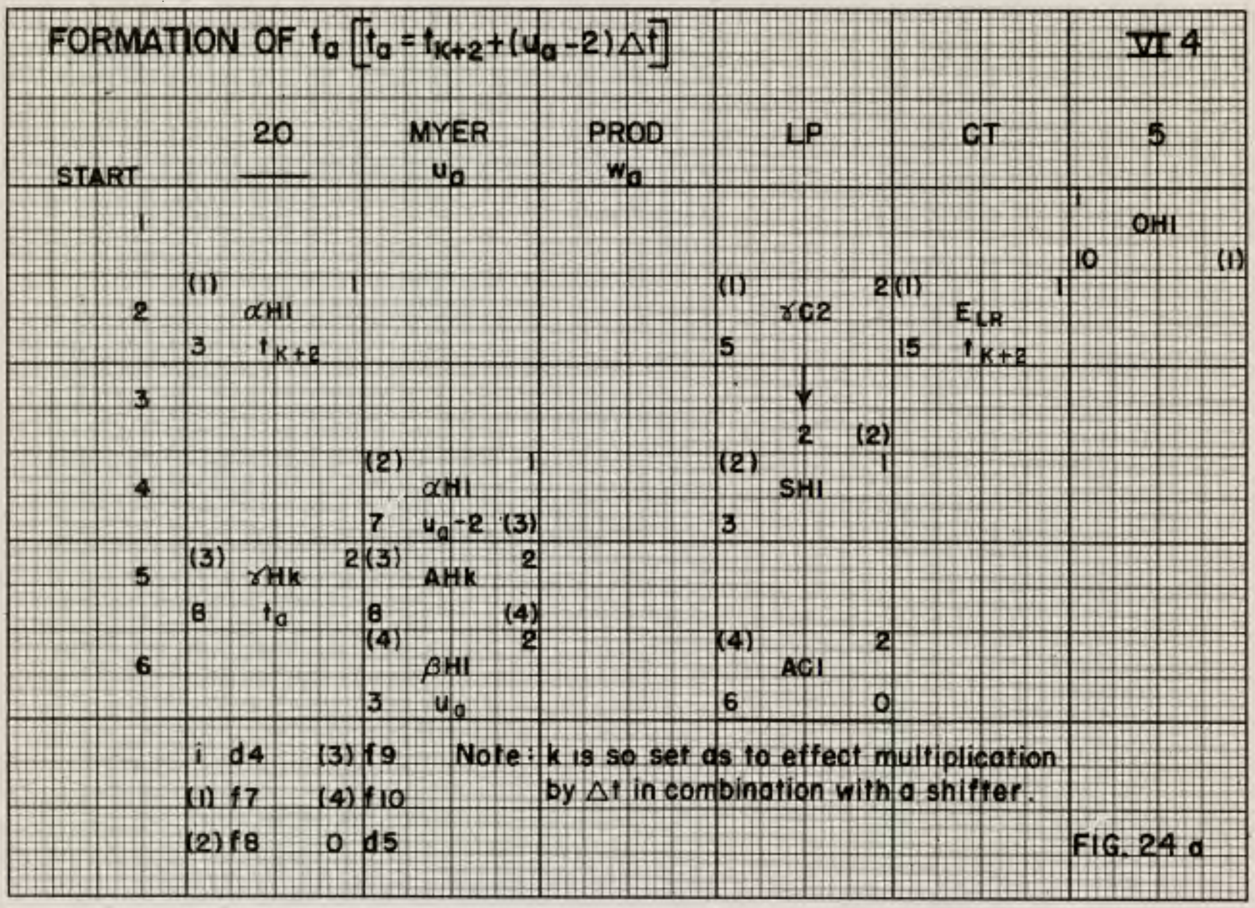
VI, VII

$\Delta^3 Z$, $\Delta^3 W$ generated in 20 by II 2

	VI	VII
i	c6	d1
(1)	h1	j1
(2)	h2	j2
(3)	h3	j3
(4)	h4	q5*
O1	b2	b2
O2	c7	d2

FIG. 23

FORMATION OF t_a $[t_a = t_{K+2} + (u_a - 2)\Delta t]$



Note: k is so set as to effect multiplication by Δt in combination with a shifter.

FIG. 24 a

FORMATION OF t_a

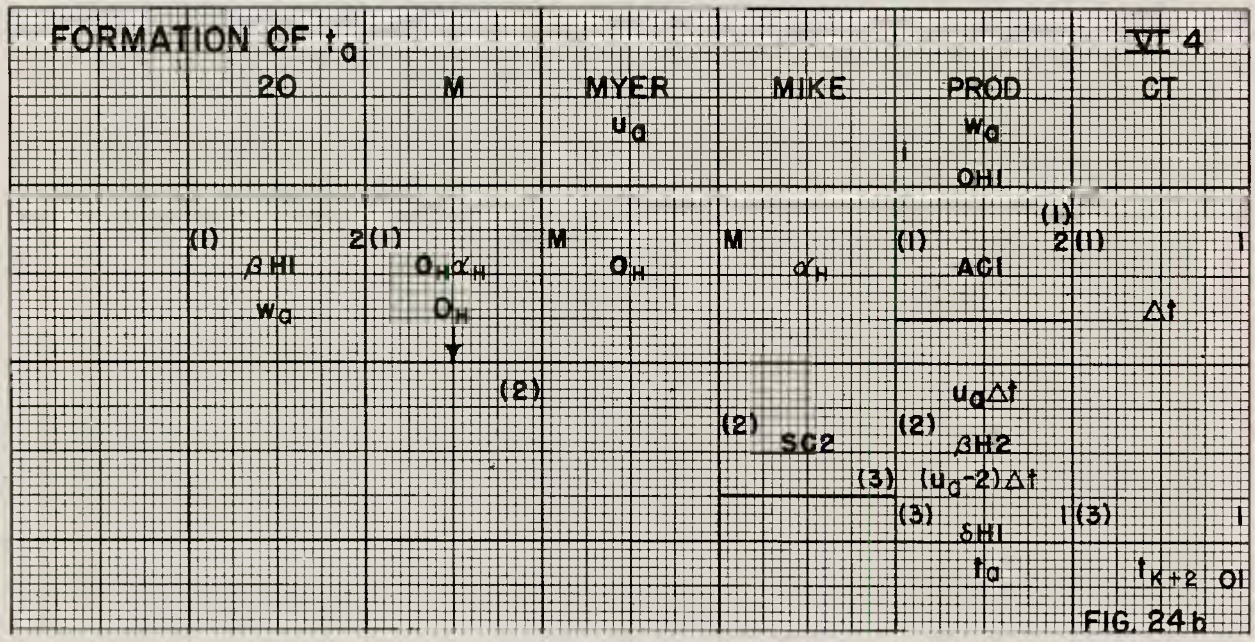


FIG. 24 b

RESTRICTED

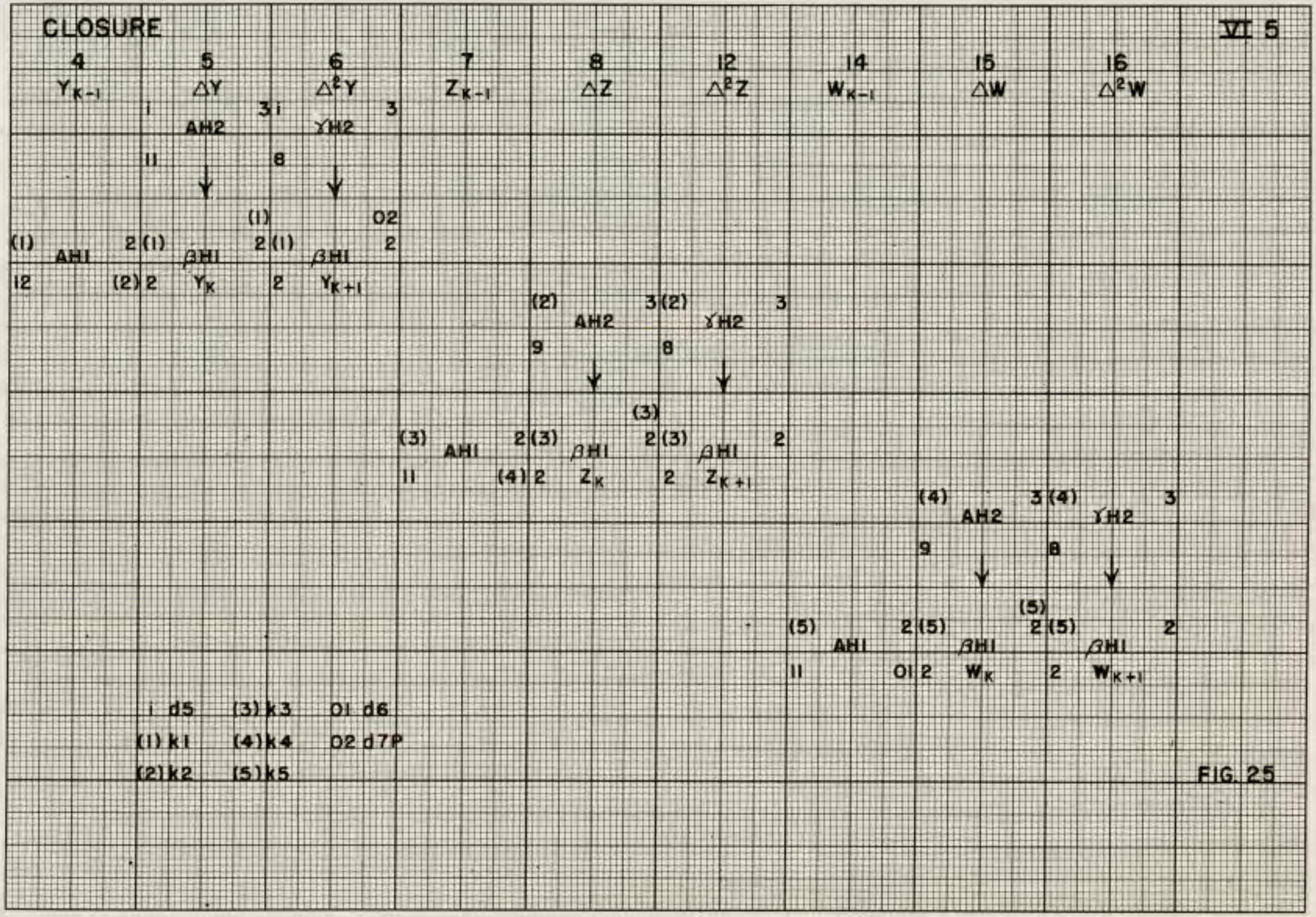


FIG. 25

RESTRICTED

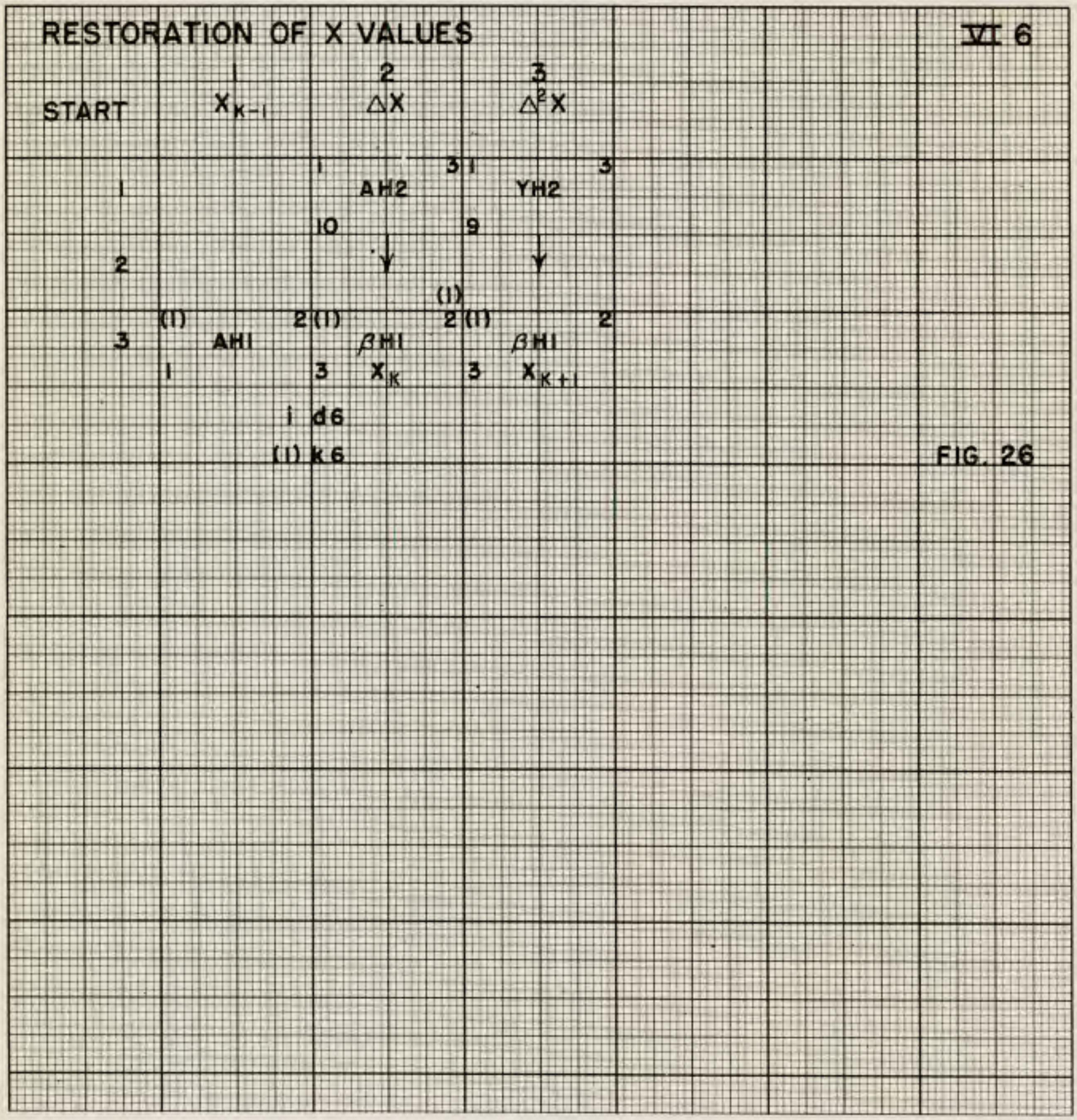
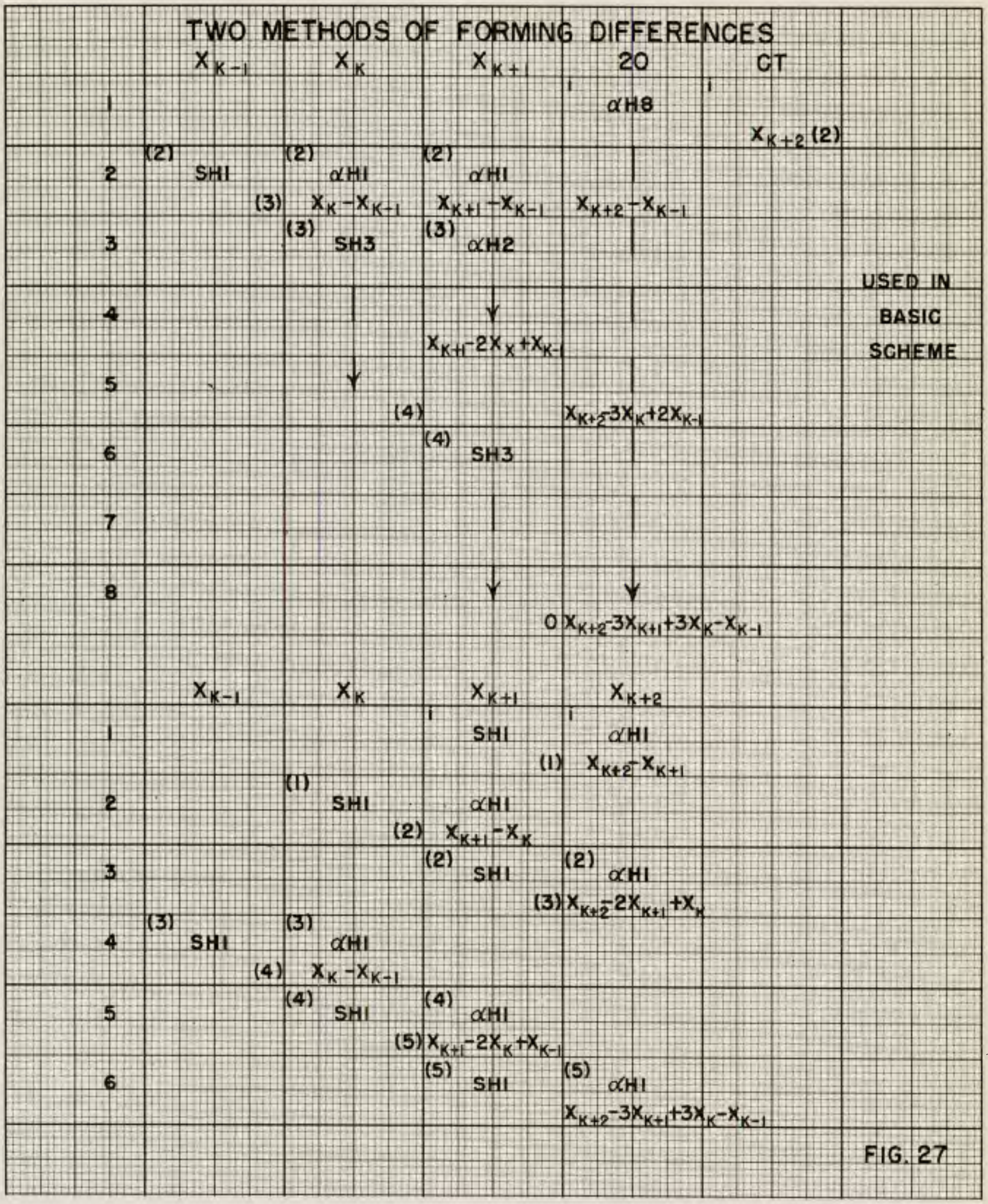

RESTORATION OF X VALUES
VI 6
START
1
X_{K-1}
2
ΔX
3
$\Delta^2 X$
1
1
AH2
3
1
YH2
3
10
9
2
(1)
(1)
2
(1)
2
(1)
2
3
AH1
βH1
βH1
1
3
X_K
3
X_{K+1}
1
d6
(1)
k6

FIG. 26



TWO METHODS OF FORMING DIFFERENCES

FIG. 27

I. CARD TESTS AND SET-UP

		0	1	2	3	4	5	6	7	8	9	10
INPUTS			a1	a2	a3	a4	a5	a6	a7	a8	a9	I.P.
CONCURRENT			I8	I8			I1			I1, I2		
ACC.	1	X_{K-1}					x X_K			x ——		
	2	X_K			○	x	x X_{K+1}			x —— a_0		
	3	X_{K+1}				x	x X_{K+2}			x —— a_0	○	
	4	Y_{K-1}					x Y_K			x ——		
	5	Y_K					x Y_{K+1}			x ——		
	6	Y_{K+1}					x Y_{K+2}			x ——		
	7	Z_{K-1}					x Z_K			x ——		
	8	Z_K			x		x Z_{K+1}		○	x ——		
MYER	9	u_0					x ——					
MIKE	10											
LP	11											
	12	Z_{K+1}					x Z_{K+2}			x ——	○	
PROD	13			x C^* —				x ——				
	14	W_{K-1}					x W_K			x ——		
	15	W_K			x		x W_{K+1}			x ——		
	16	W_{K+1}		○			x W_{K+2}			x ——		
	17a	a				x		x a + △a		x —— a_0		
	b	b		x					x b + △b			x b_0
	18	——		x C (test)		x (test)		x ——				
	19	——			x (test)			x ——				
	20	——		x φ —				x ——				
M.P.				x Σ_1	x Σ_2	x Σ_3						
C.T.				x φ_{C^*}, C_1			x X_{K+2} Y_{K+2} Z_{K+2} W_{K+2}	x △a	x △b	x a_0		x b_0
OTHER			R_i R_j									
OUTS	1		a2	a3	a4	b1	a1	a4	a8	(a2)	a8	a9
	2			a7	a5	a5	(R_1)		a2		a1	
	3			a9 or stop	a5	a6 or stop						
	4			stop								
	5			stop								

FIG. 28

		II PREPARATION FOR PRIMARY INTERPOLATION				III ITERATION							
		0	1	2		0	1	2	3	4	5	6	7
INPUTS			b1	b2			b3	b4	b5	b6	b7	b8	b9
CONCURRENT			II 2	II 1, IV 1 V 1, VI 1			III 2	III 1, IV 2 V 2, VI 2					
AGG.	1	X_{K-1}	x			X_{K-1}	x						
	2	X_K	x ΔX			ΔX	x						
	3	X_{K+1}	x Δ^2X			Δ^2X	x						
	4	Y_{K-1}				Y_{K-1}							
	5	Y_K				Y_K							
	6	Y_{K+1}				Y_{K+1}				o			
	7	Z_{K-1}				Z_{K-1}							
	8	Z_K				Z_K							
MYER	9					u_n	<	x		x u_{n+1}			
MIKE	10						<	x $6f(u_n)X$ ——					
LP	11						<	x	x $+\lambda F(u_n)X$ ——				
	12	Z_{K+1}				Z_{K+1}							
PROD.	13						<	x		x			
	14	W_{K-1}				W_{K-1}							
	15	W_K				W_K							
	16	W_{K+1}				W_{K+1}							
	17a	a				a	o		x				
	17b	b				b							
	18	——											
	19	——								x (test)			
	20	——	<	x Δ^3X		Δ^3X	<	x					
MP									x Σ4	x Σ5			
CT			x X_{K+2}										
OTHER								x M					
OUTS	1		b2	(b4)			b4		b6	c1			
	2		b3				b5		b7 or stop	b8 or stop			
	3									b3			
	4									stop			
	5									b9or stop			

FIG. 29

		IV SECONDARY INTERPOLATION					V SECONDARY INTERPOLATION		
				FOR Y			FOR Z		
		0	1	2	3	4	1	2	3
INPUTS			C1	C2	C3	C5	C6	C7	C8
CONCURRENT			II 2	III 2	IV 4, V 3 VI 3	IV 3	II 2	III 2	IV 3
ACC.	1	X_{K-1}							
	2	ΔX							
	3	$\Delta^2 X$							
	4	Y_{K-1}	x	x	0				0
	5	Y_K	x ΔY	x					
	6	Y_{K+1}	x $\Delta^2 Y$	x					
	7	Z_{K-1}					x	x	
	8	Z_K					x ΔZ	x	
MYER	9	u_a		<	← u_a	x		<	x ← u_a
MIKE	10		0	< $6Y_a$		x —		< $6Z_a$	x —
LP	11			<		x —		<	x —
	12	Z_{K+1}					x $\Delta^2 Z$	x	
PROD	13			<		x —		<	x —
	14	W_{K-1}							
	15	W_K							
	16	W_{K+1}							
	17a	a		0			0		
	b	b							
	18	—					x Y_a		
	19	—							
	20	$\Delta^3 X$	x ← < $\Delta^3 Y$		← u_a —		< $\Delta^3 Z$		← u_a —
MP									
CT			x Y_{K+2}		< 1/6		x Z_{K+2}		< 1/6
OTHER				< M		x M		< M	x M
OUTS	1		b2	b4		c6	b2	b4	d1
	2		c2	c5			c7	c8	
	3			c3				c3	
	4								
	5								

FIG. 30

VI SECONDARY INTERPOLATION FOR w AND CLOSURE

		0	1	2	3	4	5	6	7
INPUTS			d1	d2	d3	d4	d5	d6	d7
CONCURRENT			II 2	III 2	IV 3				
ACC.	1	X_{K-1}						x	
	2	ΔX						x X_K	
	3	$\Delta^2 X$						x X_{K-1}	
	4	Y_{K-1}					x		
	5	ΔY	○			○	x Y_K		
	6	$\Delta^2 Y$		○			x Y_{K+1}		
	7	Z_{K-1}					x		
	8	ΔZ					x Z_K		
MYER	9	u_a		<	x <— u_a	x			
MIKE	10	$6Z_a$		< $6W_a$	x ——				
LP	11	——		<	x ——	x			
	12	$\Delta^2 Z$					x Z_{K+1}		
PROD	13	——		<	x W_a				x
	14	W_{K-1}	x	x			x		
	15	W_K	x ΔW	x			x W_K		
	16	W_{K+1}	x $\Delta^2 W$	x			x W_{K+1}		
	17a	a							x
	b	b							x
	18	Y_a							x
	19	——	x Z_a						x
	20	——	< $\Delta^3 W$		<— u_a —	x t_a			x
MP									
CT			x W_{K+2}		< 1/6	x t_{K+2}			
OTHER				< M	x M				x P
OUTS	1		b2	b4	d4	d5	d6		d6
	2		d2	d3			$d7^P$		
	3			c3					
	4								
	5								

FIG. 31

RESTRICTED

WIRING DIAGRAM

ACCUMULATORS

PROGRAM CONTROLS	1	2	3	4	5	6	7	8	9 MYER	10 MIKE
1	K6 AH1 VI 6	f2 αH1 II 1	f2 αH1 II 1	l3 OC1 I 5	g2 αH1 IV 1	g2 αH1 IV 1	l6 OC1 I 5	h2 αH1 V 1	b6 αH1 III 4	M α III 2
2		l2 αH1 I 5	l3 αH1 I 5	l3 γH1 I 5	K1 βH1 VI 5	K1 βH1 VI 5	l7 γH1 I 5	K3 βH1 VI 5	e8 αH1 IV 3	
3		K6 βH1 VI 6	K6 βH1 VI 6		l5 γH1 I 5	l6 αH1 I 5		l8 γH1 I 5	f10 βH1 VI 4	
4		a4 SH1 I 4	g9 αH1 I 8						l4 OC1 I 5	
5	f5 OH9 III 1 p2	b3 OH9 III 1 f4	f4 AH9 III 1 f5	g5 OH9 IV 2 p5	c2 OH9 IV 2 g4	g4 AH9 IV 2 g5	h6 OH9 V 2 p8	c7 OH9 V 2 h5	c4 AC1 IV 3 e6	c1 OH1 IV 1 b2
6	p3 OH9 III 1 p4	p2 AH9 III 1 p3	f3 αH2 II 1	p6 OH9 IV 2 p7	p5 AH9 IV 2 p6	g3 αH2 IV 1	p9 OH9 V 2 p10	p8 AH9 V 2 p9	M(RS) SH1	M(DS) SH1
7	p4 OH5 III 1 f6	b1 OH1 II 1 b2	p1 SH3 II 1 b3	p7 OH5 IV 2 g6	g6 OH4 IV 2 c3	q4 SH3 IV 1 c2	p10 OH5 V 2 h7	h7 OH4 V 2 c3	f8 αH1 VI 4 f9	e3 γH6 III 2 e4
8	f6 AH6 III 1 b5	f3 SH3 II 1 p1	l2 AC1 I 5 l3	g6 AH6 IV 2 c5	g3 SH3 IV 1 q4	d5 γH2 VI 5	h7 AH6 V 2 c8	h3 SH3 V 1 h4	f9 AHk VI 4 f10	e5 βH7 III 2
9	b1 OH1 II 1 f1	l1 AC1 I 5 l2	d6 γH2 VI 6	g2 SH1 IV 1 g3		l5 AC1 I 5 l6	c6 OH1 V 1 b2	K2 AH2 VI 5 K3	n7 AH1 III 4 n3	b10 ACk_1 III 3 b6
10	f2 SH1 II 1 f3	d6 AH2 VI 6 K6	m2 OH1 I 4 m4	e7 OH6 IV 3 q6	d4 OH1 VI 4 f7		h2 SH1 V 1 h3	l7 AC1 I 5 l8		b5 αH6 III 3
11	a5 OC1 I 5 l1	m7 OH1 I 3 m9	m5 SH1 I 4 m1	q6 OH6 IV 3 e8	d5 AH2 VI 5 K1	n2 OH1 III 4 n3	K3 AH1 VI 5 K4	m6 AH3 I 3 m8		
12	l1 γH1 I 5 a1	g9 αH1 I 8	a9 OH1 I 9 a1	K1 AH1 VI 5 K2	l4 AC1 I 5 l5	d2 OH1 VI 2 b4		d9 OH1 I 7 a2		

FIG. 32

RESTRICTED

WIRING DIAGRAM

FIG. 32

ACCUMULATORS

PROGRAM CONTROLS	11 LP	12	13 PROD	14	15	16	17	18	19	20
1	M(F) AC	h2 αH1 V1	M βH	l9 OC1 I5	j2 αH1 VI1	j2 αH1 VI1	m1 AH1 I4	d2 αH1 I2	n1 SH1 III4	c4 βH1 IV3
2	M(RS) βH	K3 βH1 VI5	M βH	l10 γH1 I5	K5 βH1 VI5	K5 βH1 VI5	a6 αH1 I6	K7 εC1 I2	q8 SC1 I3	e8 AC1 IV3
3	f8 SH1 VI4	l9 αH1 I5	e2 SC1 III2	g7 βH1 I8	c9 γH1 I5	c10 αH1 I5	g9 αH1 I8	d8 OC1 I6	n8 δH1 III4	f7 αH1 VI4
4	b6 SC1 III4		e4 SC1 III2	g8 AC1 I8			a10 βH1 I10		a3 αH1 I3	d8 OC1 I6
5	f7 γG2 VI4 f8	h5 AH9 V2 h6	e1 γH3 III2 e2	j5 OH9 VI2 q1	d2 OH9 VI2 j4	j4 AH9 VI2 j5	h10 SH1 I2 j9	c6 βH1 V1 h1	d1 βH1 VI1 j1	b2 αH8 II2
6	f10 AC1 VI4 d5	h3 αH2 V1	n7 εG2 III4 n8	q2 OH9 VI2 q3	q1 AH9 VI2 q2	j3 αH2 VI1	b5 SH6 III3	j9 αH1 I2 j10	b6 εH1 III4 n1	c1 OC1 IV1 g1
7	b10 βHk$_1$ III3	h4 SH3 V1 c7	n8 SC1 III4 n5	q3 OH5 VI2 j6	j6 OH4 VI2 c3	q5 SH3 VI1 d2	b3 OH1 III1 b4	j10 AH1 I2 K9	n4 εC1 III4 n5	c3 OC1 IV3 c4
8		K2 γH2 VI5	j7 αC1 I2 j9	j6 AH6 VI2 d3	j3 SH3 VI1 q5	K4 γH2 VI5	c2 OH1 IV2 b4	a4 αH1 I4 m1	n7 βH1 III4 n1	f9 γHk VI4
9		l8 AC1 I5 l9	j8 SC1 I2	d1 OH1 VI1 b2	K4 AH2 VI5 K5	c9 AC1 I5 c10	c7 OH1 V2 b4	m1 γH1 I4 q7	m6 γH2 I3 q8	h10 γC1 I2 e9
10		a9 OH1 I9 a8	d8 OC1 I6 a4	j2 SH1 VI1 j3	l10 AC1 I5 c9	K7 OH1 I2 K8	a7 βH1 I7 d9	m5 αH1 I4 m4	n5 SC1 III4 n6	e9 SC1 I2 K8
11			b6 OH2 III4 n4	K5 AH1 VI5 d6	a3 SH1 I3 m6	K9 OH1 I2 K10	g7 AC1 I8 g8	q7 SC1 I4 m3	d8 OC1 I6	b4 AH1 III2
12			M AC1 III2		m10 AH1 I3 m6		g8 γH1 I8 g8	K10 AC1 I2 h8	m10 γH1 I3 m9	h9 αH1 I2 h10

WIRING DIAGRAM

PROGRAM CONTROLS

FIG. 33a

		1	2	3	4	5	6	7	8	9	10
Multiplier	1-10	b4 III 2 $O_H\alpha_H$ 9 O e1	e2 III 2 $O_H\alpha_C$ 9 O e3	e4 III 2 $O_H\alpha_H$ 9 AC e5	c5 IV 4 $\alpha_C O_C$ 9 AC c6	c8 V 3 $\alpha_C O_C$ 9 AC d1	d3 VI 3 $\alpha_C O_C$ 9 OH d4				
	11-20										
	21-24										
Constant Transmitter	1-10	f1 II 1 A_{LR} f2	g1 IV 1 B_{LR} g2	l3 I 5 A_{LR} l4	l6 I 5 B_{LR} l7			h1 V 1 G_{LR} h2	j1 VI 1 D_{LR} j2	l9 I 5 G_{LR} l10	c10 I 5 D_{LR}
	11-20			h9 I 2 F_R K8	a2 I 2 F_L j7	f7 VI 4 E_{LR}				j7 I 2 G_L j8	a7 I 7 H_R a8
	21-30	a6 I 6 H_L d8				e6 IV 3 K_{LR} e7	a10 I 10 J_R a9	g9 I 8 J_L			

DIGIT CONNECTIONS

	1	2	3	4	5	6	7	8	Myer 9	Mike 10	LP 11	12	Prod 13	14	15	16	17	18	19	20
α		1	1		1	1		1	1	1	Sp M	1	Sp M		1	1	1 shifter	1	1	1
β		2	2		2	2		2	2	2	1 M	2	2 M	3 shifter	2	2	1 shifter	2	2	2
γ	3	3	3	3	3	3	3	3		3	2 shifter	3	3	3	3	3	2	3 shifter	3	2 shifter
δ		3 shifter	3 shifter							2 shifter			1						1 Shifter	
ϵ											M = permanent connections								1 Shifter	
A	2	3	3	2	3	3	2	3	2	1	2 M	3	2	2	3	3	3	PM to K7	PM to n2	1
S	1	1	1	1	1	1	1	1	1 M	2 M	1	1	1	1	1	1	2	PM to m2	PM to m7	1

WIRING DIAGRAM

STEPPERS

	Reg.	Di	Dcl	1	2	3	4	5	6
Σ_1	h8	k8	a2	—	h9	—	a9	a3	a7
							or —		
Σ_2	m8	m9	a3	a5	m10	a5	a4		
Σ_3	m3	m4	d4	a6	m5	b1	a5		
				or —					
Σ_4	b5			b10	b7				
					or —				
Σ_5	n6	n3	b6	b8	c1	n7	—	b3	b9
				or —					or —

INITIATING UNIT

R_i	R_l	R_o	C_i	C_o	P_i	P_o	IP
a1	a1	a2	a8	—	d7	a6	a10

FIG. 33 b

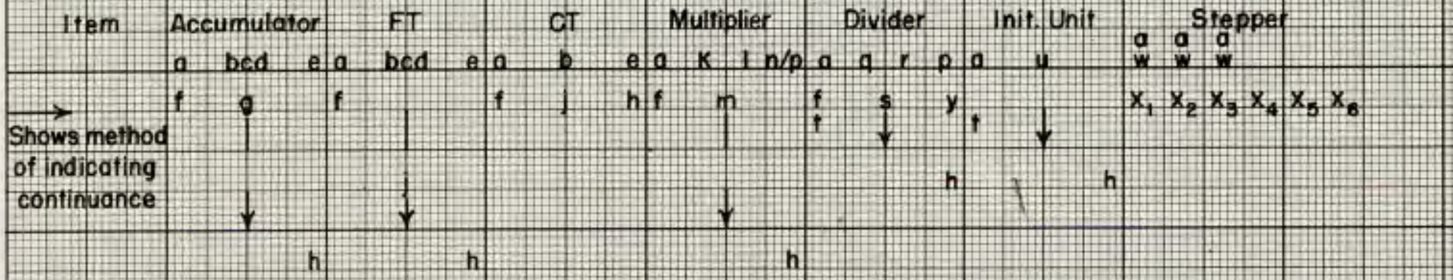
PROGRAM TRAYS

		1	2	3	4	5	6	7	8	9	10	X
	a	I 1 (1)	I 2 (1)	I 3 (1)	I 4 (1)	I 5 (1)	I 6 (1)	I 7 (1)	I 8 (1)	I 9 (1)	I 10 (1)	
	b	II 1 (1)	II 2 (1)	III 1 (1)	III 2 (1)	III 3 (1)	III 4 (1)	[III 5 (1)]	[III 6 (1)]	[III 7 (1)]	I. P. III 3 (1)	
	c	IV 1 (1)	IV 2 (1)	IV 3 (1)	IV 3 (1)	IV 4 (1)	V 1 (1)	V 2 (1)	V 3 (1)	I 5 (11)	I 5 (12)	
	d	VI 1 (1)	VI 2 (1)	VI 3 (1)	VI 4 (1)	VI 5 (1)	VI 6 (1)	VI 7 (1)	I 6 (1)	I 7 (1)		
	e	III 2 (1)	III 2 (2)	III 2 (3)	III 2 (4)	III 2 (5)	IV 3 (2)	IV 3 (3)	IV 3 (5)	I 2 (12)		
	f	II 1 (1)	II 1 (2)	II 1 (3)	III 1 (1)	III 1 (2)	III 1 (6)	VI 4 (1)	VI 4 (2)	VI 4 (3)	VI 4 (4)	
	g	IV 1 (1)	IV 1 (2)	IV 1 (3)	IV 2 (1)	IV 2 (2)	IV 2 (6)	I 8 (1)	I 8 (2)	I 8 (3)		
	h	V 1 (1)	V 1 (2)	V 1 (3)	V 1 (4)	V 2 (1)	V 2 (2)	V 2 (6)	I 2 (9)	I 2 (10)	I 2 (11)	
	j	VI 1 (1)	VI 1 (2)	VI 1 (3)	VI 2 (1)	VI 2 (2)	VI 2 (6)	I 2 (1)	I 2 (2)	I 2 (3)	I 2 (4)	
	k	VI 5 (1)	VI 5 (2)	VI 5 (3)	VI 5 (4)	VI 5 (5)	VI 6 (1)	I 2 (5)	I 2 (6)	I 2 (7)	I 2 (8)	
	l	I 5 (1)	I 5 (2)	I 5 (3)	I 5 (4)	I 5 (5)	I 5 (6)	I 5 (7)	I 5 (8)	I 5 (9)	I 5 (10)	
	m	I 4 (1)	I 4 (3)	I 4 (4)	I 4 (5)	I 4 (6)	I 3 (1)	I 3 (3)	I 3 (4)	I 3 (5)	I 3 (6)	
	n	III 4 (1)	III 4 (2)	III 4 (3)	III 4 (4)	III 4 (5)	III 4 (6)	III 4 (7)	III 4 (8)			
	o											
	p	II 1 (4)	III 1 (3)	III 1 (4)	III 1 (5)	IV 2 (3)	IV 2 (4)	IV 2 (5)	V 2 (3)	V 2 (4)	V 2 (5)	
Jumpers	q	VI 2 (3)	VI 2 (4)	VI 2 (5)	IV 1 (4)	VI 1 (4)	IV 3 (4)	I 4 (2)	I 3 (2)			
	r											

FIG. 34

CODE FOR PROGRAMMING CONVENTIONS FOR ENIAC

PROGRAMMING CONVENTIONS



a Number of input program line
b Function or selective switch setting
c Clear switch setting
d Repeat switch setting
e Digit trunk or trunks
f Program control number
g Description of quantity held at end of indicated add times
h Output program line (at time it occurs)
j Description of quantity emitted (when first emitted)
k Myer switch (with clear indication)
l Mike switch (with clear indication)
m Product disposal switch
n Significant figure switch

p Places switch
q Numerator accumulator switch
r Denominator accumulator switch
s Answer disposal and interlock switch
t Reference for interlock
u Type of program
R reader P printer C selective clear
I initiating pulse
v Stepper number
w Type of input
R regular; Dor Di direct input;
Cor Dcl direct clear, E_k input to k th decade
X_k Outputs from positions k of stepper
y Round off switch

FIG. 35

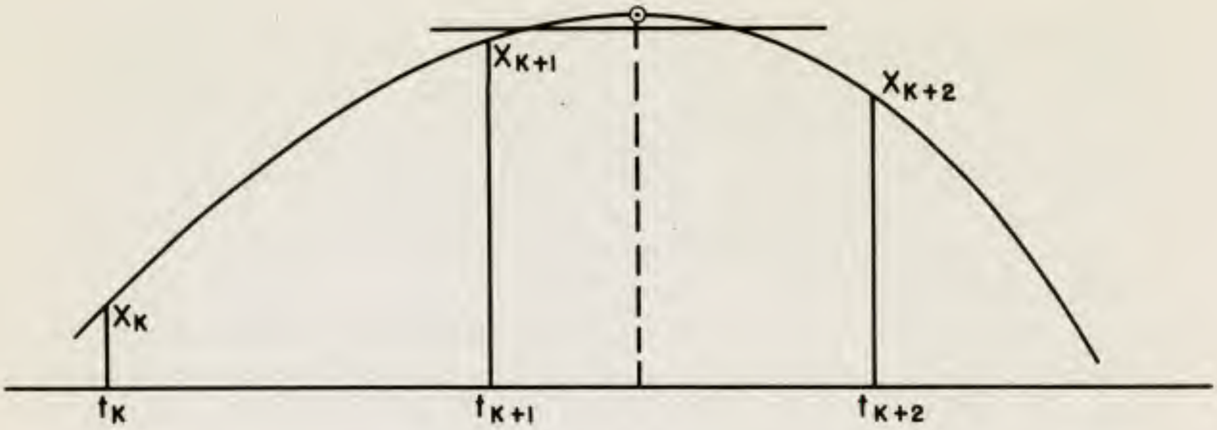


FIG. 36

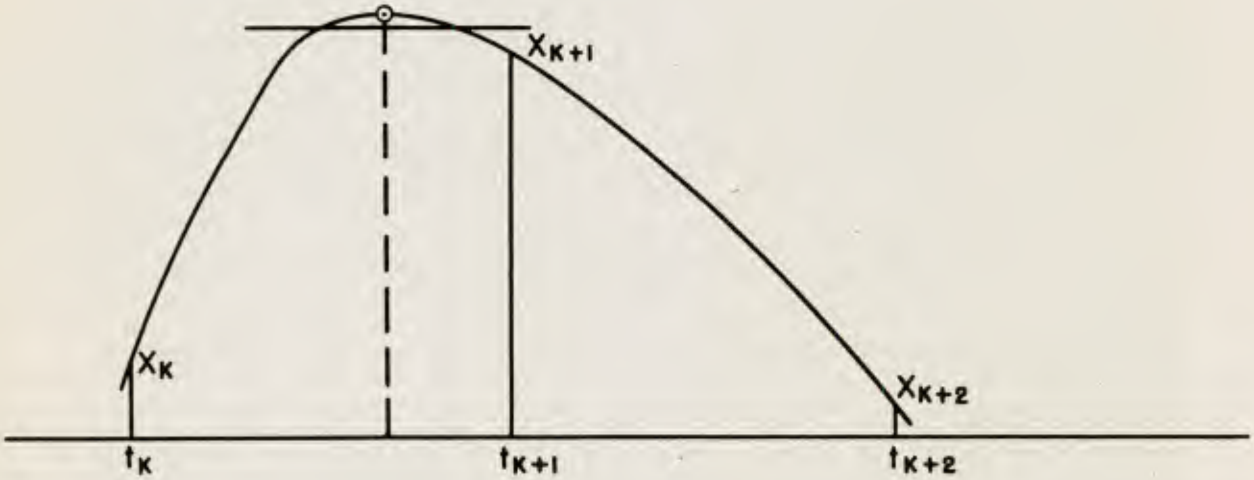


FIG. 37

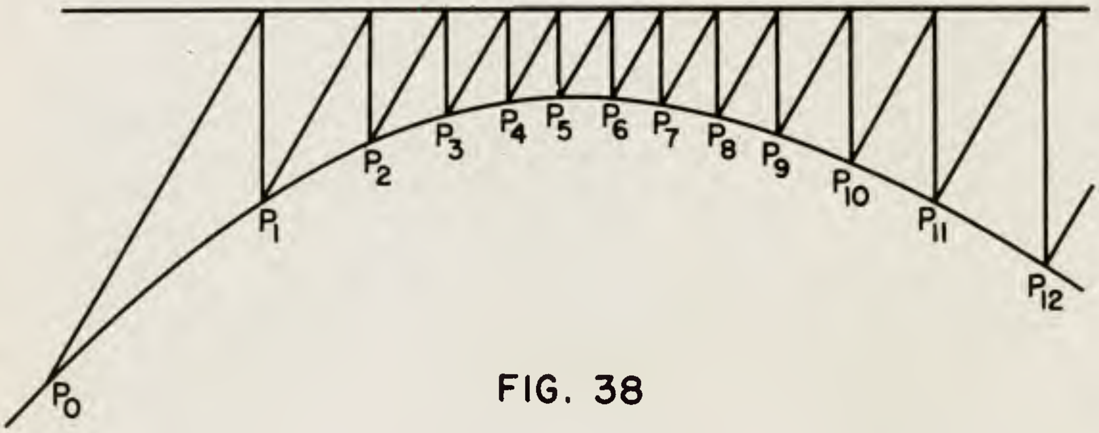


FIG. 38

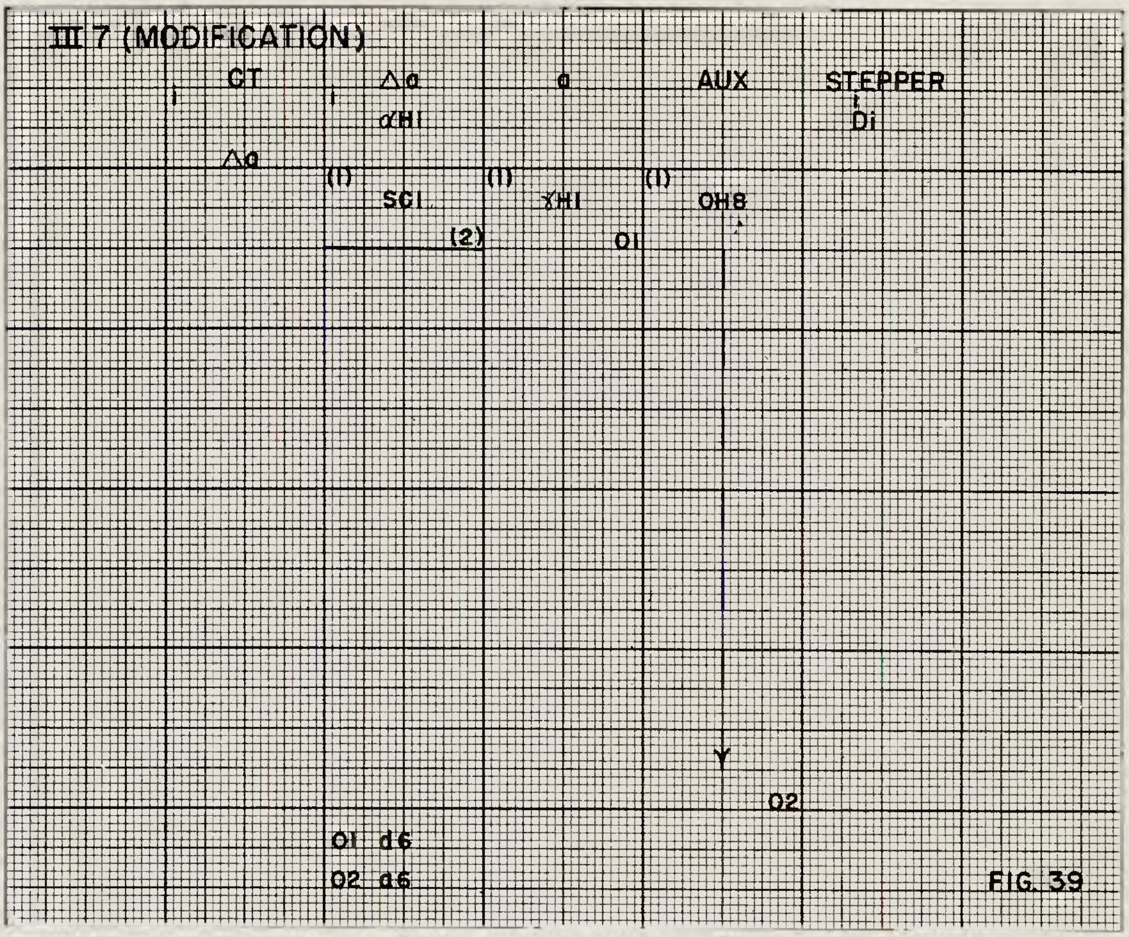

III 7 (MODIFICATION)
i
CT
i
Δa
a
AUX
STEPPER
αHI
Di
Δa
(1)
SC1
(1)
γHI
(1)
OH8
(2)
O1
O2
O1 d6
O2 d6

FIG. 39

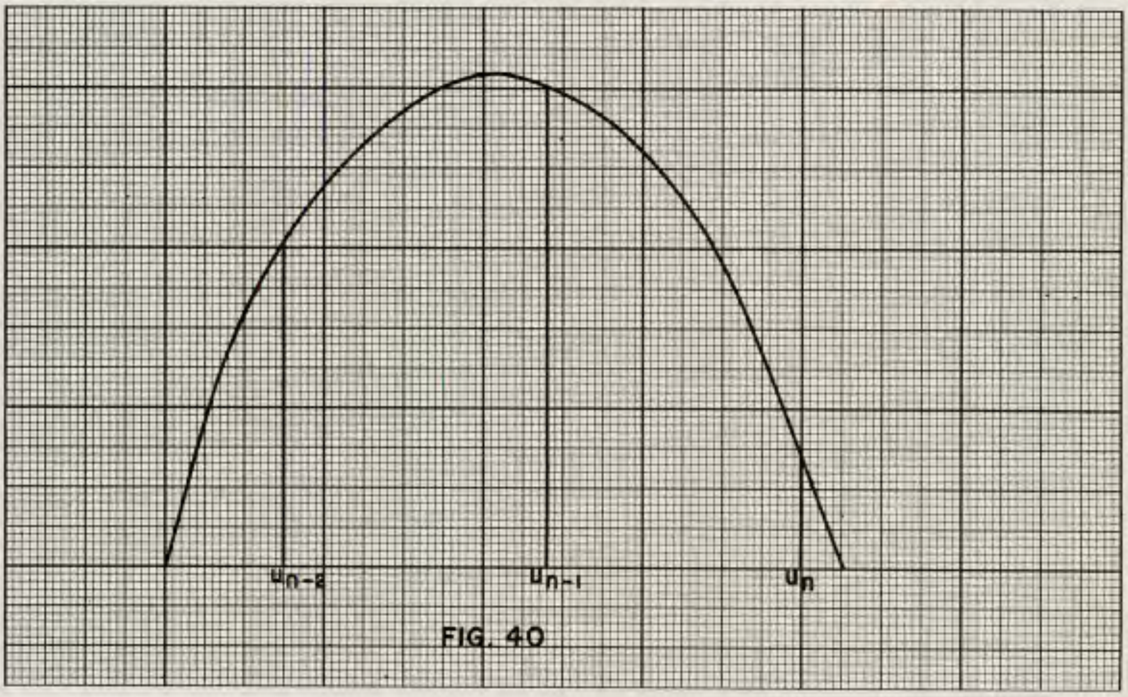

u_{n-2}
u_{n-1}
u_n

FIG. 40

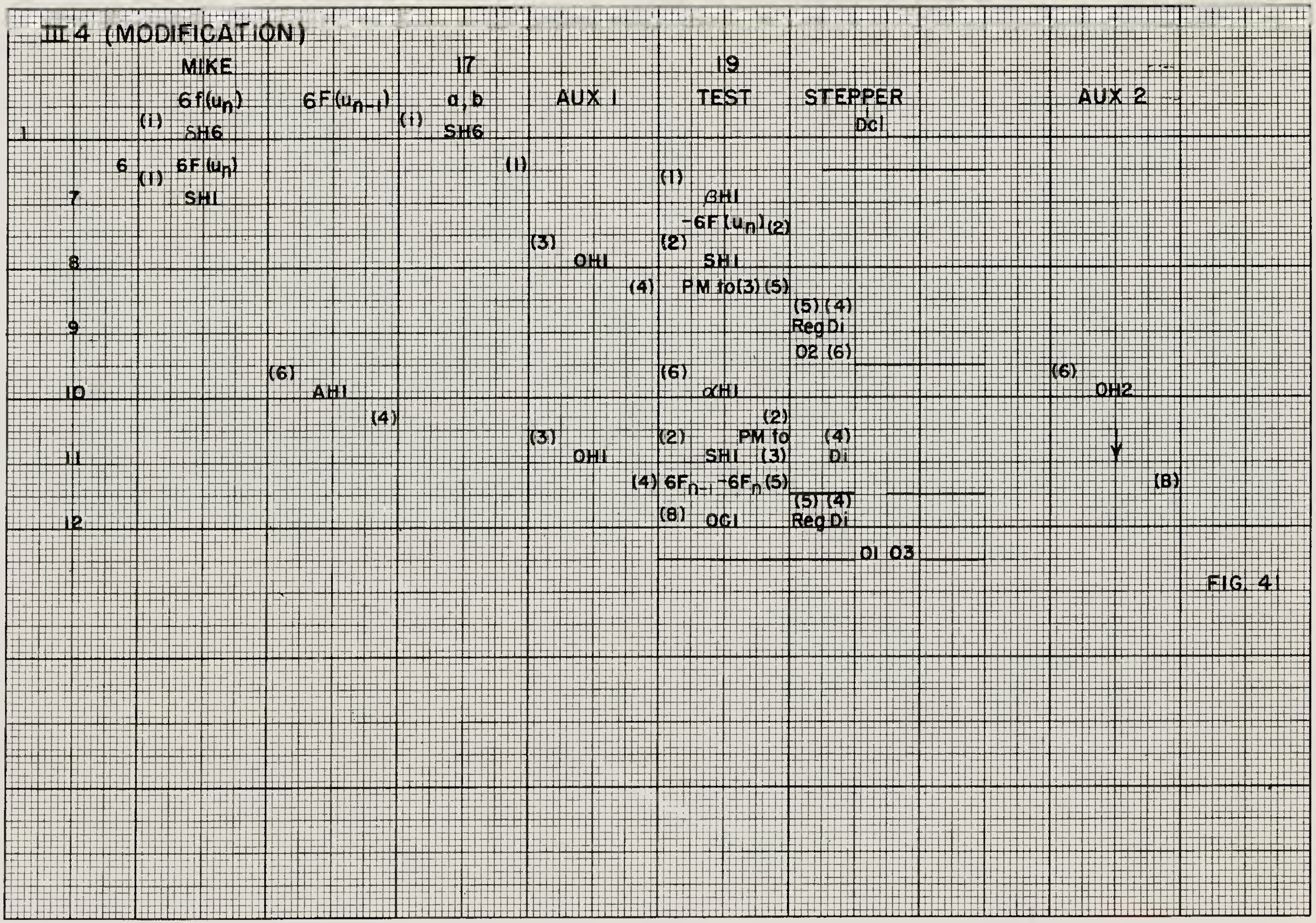
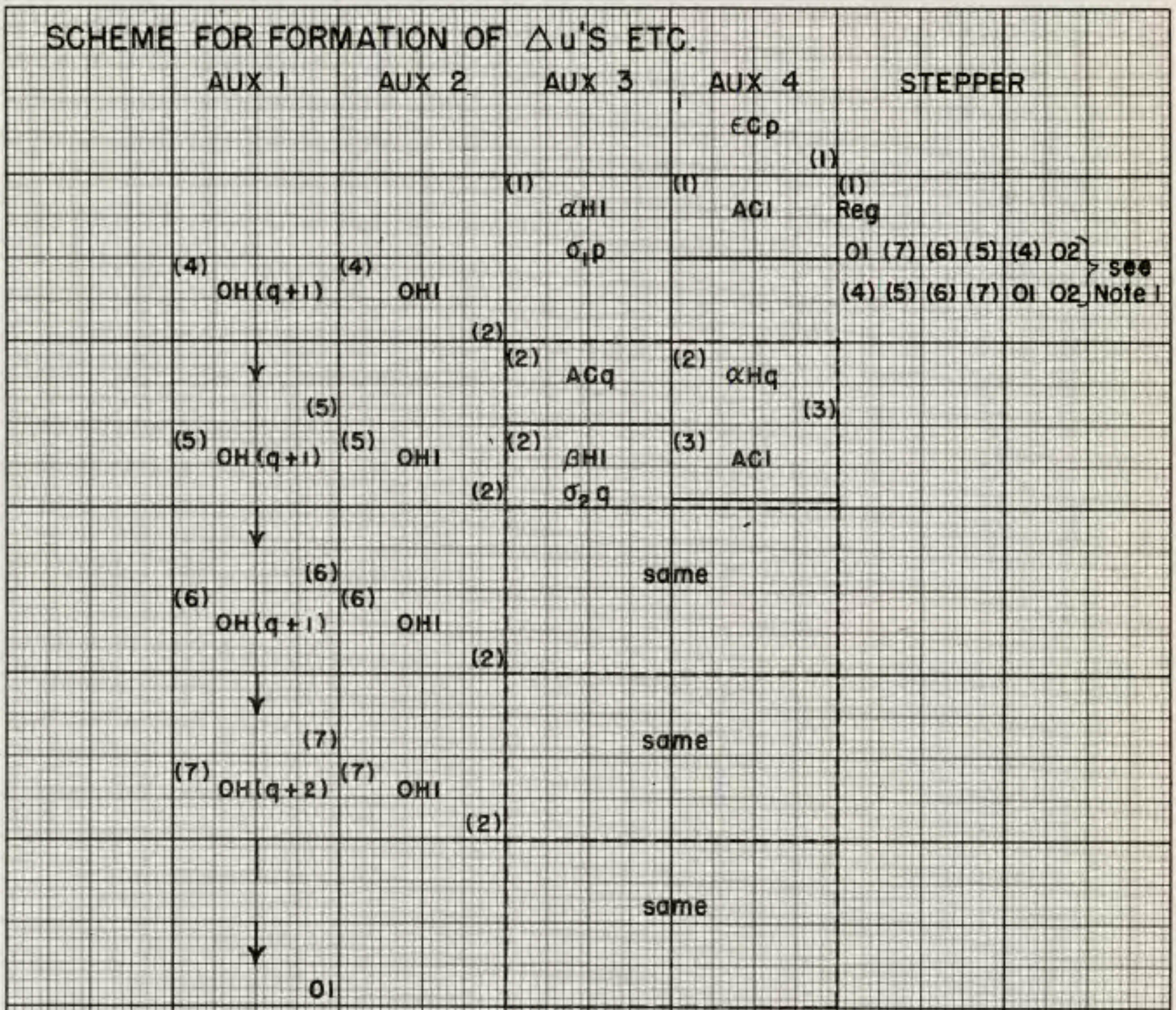

III 4 (MODIFICATION)
MIKE
17
19
6f(u_n)
6F(u_{n-1})
a, b
AUX 1
TEST
STEPPER
AUX 2
1
(i)
δSH6
(i)
SH6
Dcl
6
(1)
6F(u_n)
(1)
(1)
7
SHI
βHI
-6F(u_n)
(2)
(3)
(2)
8
OHI
SHI
(4)
PM to (3)
(5)
(5)
(4)
9
Reg Di
O2
(6)
(6)
(6)
(6)
10
AHI
αHI
OH2
(4)
(2)
(3)
(2)
PM to
(4)
11
OHI
SHI
(3)
Di
(4)
6F_{n-1}
-6F_n
(5)
(8)
(5)
(4)
12
(8)
OCI
Reg Di
O1 O3

FIG. 41

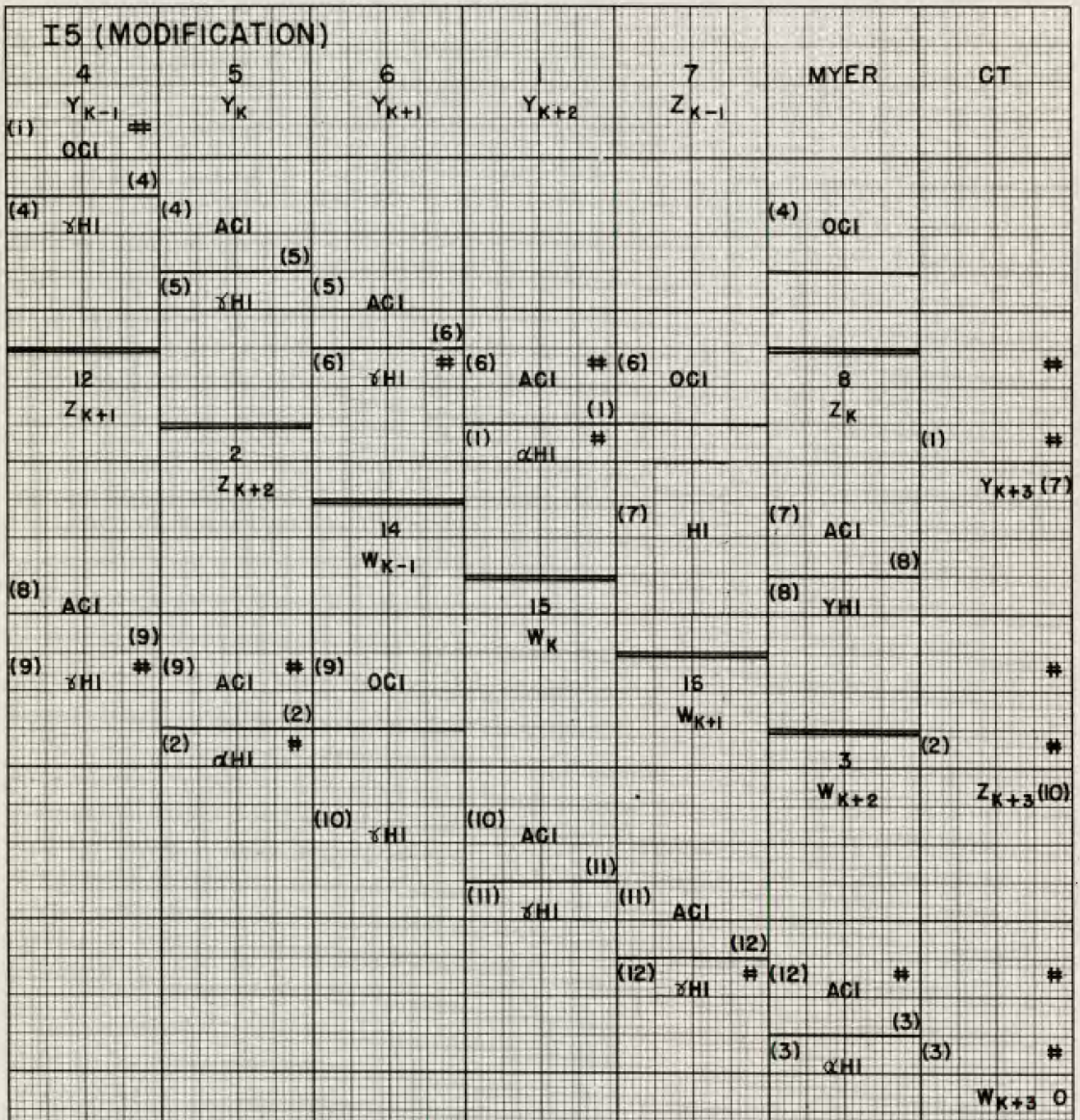


Note: Let $\Delta_K u$ be the amount left in AUX 3 with stepper in kth position.

Let σ_1 and σ_2 be powers of 10 such that the shifters in (1) and (2), (3) multiply by σ_1 and σ_2 respectively, then

$$\Delta_K u = \begin{cases} \sigma_1 p(\sigma_2 q)^{K-1} & \text{using first row of stepper outputs} \\ \sigma_1 p(\sigma_2 q)^{5-K} & \text{using 2nd row of stepper outputs} \end{cases}$$

FIG. 42



This is the same as I5 except for changes in accumulators 1,2,3 and in items marked #.

FIG. 43

FLOW CHART

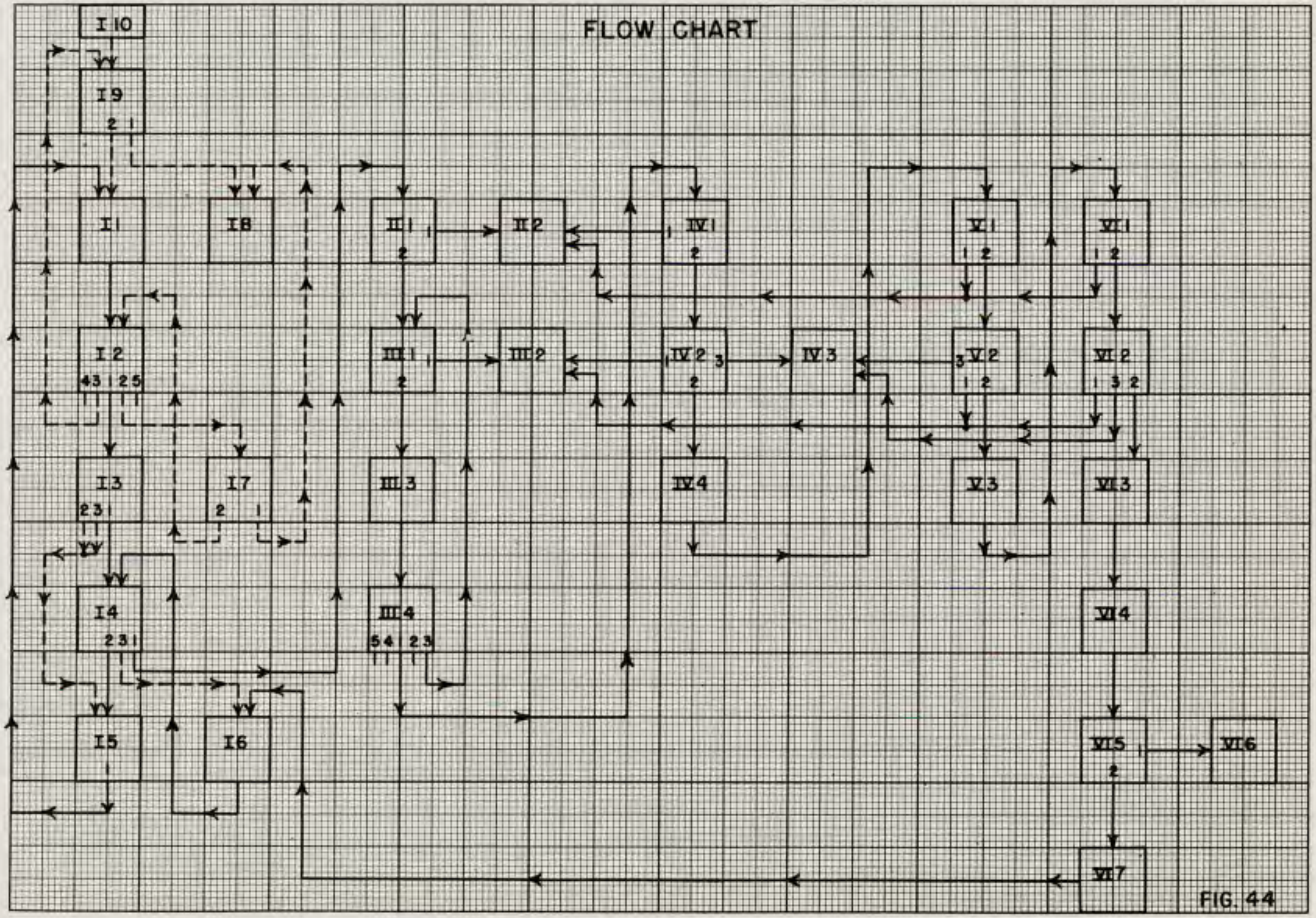


FIG. 44